Ingénierie cryptographique :
Aspects sécuritaires et algorithmiques

Marc Joye

Mémoire soumis pour l'habilitation à diriger des recherches (HDR), spécialité informatique.

Jury : Bernard Coulette, U. Toulouse II
Jean-Marc Couveignes, U. Toulouse II (directeur)
Yves Deswarte, LAAS-CNRS
Marc Girault, FT R&D (rapporteur)
Antoine Joux, DCSSI
Refik Molva, Eurécom
François Morain, École Polytechnique
Christof Paar, Ruhr-U. Bochum (rapporteur)
Jean-Jacques Quisquater, UCL
Jacques Stern, ENS (président)
Serge Vaudenay, EPFL
Moti Yung, Columbia U. (rapporteur)

Septembre 2003

À Hung-mei & Juliette.

«*L'art d'enseigner n'est que l'art d'éveiller*
la curiosité des jeunes âmes.»

— Anatole France

Préface

Au cours de ces dernières années, j'ai eu la chance de rencontrer de nombreuses personnes exceptionnelles tant sur le plan scientifique que sur le plan humain. J'ai découvert la cryptographie en 1993 par un cours de Jean-Jacques Quisquater, mon «mentor». Malgré un horaire surchargé et souvent bousculé, Jean-Jacques a toujours su prêter une oreille attentive à mes travaux. Parmi mes autres heureuses rencontres, j'ai eu le plaisir de croiser le chemin de Jean-Marc Couveignes. Dès 1994, Jean-Marc m'a fait découvrir la théorie des courbes elliptiques, d'abord lors d'un cours donné conjointement avec François Morain, ensuite lors de séances de travail à l'ENS (merci à Jacques Stern d'avoir rendu ces visites possibles) puis à Eindhoven. Après ma thèse, je suis parti près de deux ans à Taiwan où j'ai fait la connaissance de Sung-Ming Yen, lequel m'a fait partager son enthousiasme pour la recherche dans des domaines variés. Depuis septembre 1999, je travaille pour Gemplus dans l'équipe de Philippe Proust, elle-même dirigée par David Naccache. Philippe m'a laissé une certaine liberté dans mes travaux de recherche et David a su créer un environnement propice à cette recherche. L'entrée dans le monde industriel m'a cependant fait comprendre l'importance des enjeux applicatifs et économiques liés à la recherche.

Par ailleurs, je remercie tous mes collègues (trop nombreux pour être cités) du groupe CRYPTO de l'UCL, du LiS à Taiwan et du département STD de Gemplus. Merci également à tous mes co-auteurs : Giuseppe Ateniese, Feng Bao, Alexis Bernard, Olivier Benoît, Olivier Billet, Daniel Bleichenbacher, Jean-Marc Boucqueau, Éric Brier, Jan Camenisch, Junghee Cheon, Benoît Chevallier-Mames, Mathieu Ciet, Christophe Clavier, Jean-Sébastien Coron, Nicolas Degand, Jean-François Delaigle, Robert H. Deng, Jean-François Dhem, Nathalie Feyt, Aurore Gillet, Helena Handschuh, Tzonelih Hwang, Seungjoo Kim, François Koeune, Pao-Yu Ku, Narn-Yih Lee, Arjen K. Lenstra, Seongan Lim, Benoît Macq, Masahiro Mambo, Henri Massias, Patrick Mestré, David Naccache, Francis Olivier, Pascal Paillier, Richard Pinch, David Pointcheval, Stéphanie Porte, Jean-Jacques Quisquater, Tsuyoshi Takagi, Gene Tsudik, Christophe Tymen, Serge Vaudenay, Karine Villegas, Dongho Won, Sung-Ming Yen, Moti Yung et Yuliang Zheng.

Je suis également reconnaissant à Bernard Coulette, Jean-Marc Couveignes, Yves Deswartes, Marc Girault, Antoine Joux, Refik Molva, François Morain, Christof Paar, Jean-Jacques Quisquater, Jacques Stern, Serge Vaudenay et Moti Yung de me faire le plaisir de participer à ce jury d'habilitation. Je suis honoré d'avoir pu réunir la quasi totalité des meilleurs cryptographes français. Je suis également honoré d'avoir comme rapporteurs des experts de renommée internationale dans leurs domaines respectifs : Marc Girault, Christof Paar et Moti Yung. Je les remercie vivement pour leur travail de relecture. Enfin et surtout, je tiens à remercier Jean-Marc Couveignes pour tous ses efforts relatifs à la soutenance de cette habilitation à l'Université de Toulouse II.

✎ Marc Joye

Table des matières

Chapitre 1

Introduction

La cryptographie est l'art de pouvoir communiquer des informations secrètes de façon sécurisée. Bien qu'elle remonte à des temps immémoriaux, la cryptographie en temps que science (et non qu'art) est relativement récente. L'événement déclencheur a été la parution de l'article *'New directions in cryptography'* en 1976 par Diffie et Hellman [52]. Depuis, la cryptographie n'est plus le domaine réservé des militaires et a été reprise par le monde académique pour devenir une science à part entière. En plus de la confidentialité, la cryptographie s'est enrichie de nombreuses autres applications telles la signature numérique ou l'authentification.

La communauté cryptographique est très active et ne cesse de croître. Sa richesse réside dans sa diversité. Un bon cryptographe doit avoir des compétences dans des disciplines aussi variées que la théorie des nombres ou la micro-électronique. Mais les qualités premières d'un cryptographe (comme de tout chercheur) sont la curiosité et l'impétuosité : il ne faut pas avoir peur de sortir des sentiers battus et mettre en doute les idées reçues. Toutes les découvertes majeures en cryptographie répondent à cette définition. On notera à titre d'exemple la cryptographie à clé publique, les preuves à apport de connaissance nul ou encore les attaques par canaux cachés.

Sans avoir la prétention d'introduire des résultats majeurs, ce mémoire illustre certaines contre-vérités, notamment

- un cryptosystème non-homomorphique n'est pas sujet aux attaques de type multiplicatif ;
- l'utilisation de courbes elliptiques accroît la sécurité d'un cryptosystème ;
- les paddings de la norme PKCS #1 v1.5 assurent la sécurité du RSA en chiffrement et en signature ;
- un schéma de chiffrement à sécurité prouvée IND-CCA2 résiste aux attaques à messages chiffrés choisis ;
- un schéma de signature de groupe n'est efficace que si l'on suppose l'absence de collusion ;
- il est dangereux d'utiliser le même jeu de clés ou le même padding pour la signature et le chiffrement ;
- s'il existe, le recodage gauche-droite en représentation signée ne peut pas être optimal ;
- les algorithmes de génération de nombres premiers sont à l'état de l'art ;
- les protections contre les attaques à canaux cachés impliquent une dégradation non-négligeable des performances ;
- les formules pour l'addition et le doublement de points sur courbe elliptique sont nécessairement différentes ;

– les paramètres cryptographiques publics ne doivent pas être protégés.

La plupart de ces résultats est liée à mon activité d'ingénieur, à savoir le développement d'applications sécurisées. Les travaux de cryptanalyse sont le fruit de l'étude de systèmes cryptographiques alors que les travaux de cryptographie (non seulement les algorithmes ou les protocoles mais également leur implantation sécurisée) répondent à des problèmes concrets ou soulevés.

Organisation

Le reste de ce mémoire est divisé en quatre parties principales. La première partie (chapitre 2) reprend les résultats de cryptanalyse de divers schémas cryptographiques. La deuxième partie (chapitre 3) présente de nouvelles constructions cryptographiques. Les troisième et quatrième parties (chapitres 4 et 5) traitent respectivement d'implantations efficaces et sécurisées.

Quelques travaux choisis sont donnés dans leur intégralité dans l'annexe A. Enfin, l'annexe B présente une courte biographie de l'auteur.

Crédits

Comme il est coutumier dans la communauté cryptographique, la plupart des résultats présentés dans ce mémoire ont été obtenus en collaboration avec d'autres chercheurs. Leurs noms sont précisés par les références bibliographiques jalonnant le texte.

Chapitre 2

Cryptanalyse

2.1 Cryptosystèmes de type RSA

Le cryptosystème ou plus précisément la primitive cryptographique la plus utilisée est sans conteste le RSA [140]. La sécurité de cette primitive a été très largement étudiée. La plupart des attaques connues à ce jour sont reprises dans [18].

Plus récemment, d'autres structures ont été considérées pour implanter des analogues au RSA, notamment les suites de Lucas [147] et les courbes elliptiques [112, 50].

Bien qu'il soit maintenant compris que la primitive RSA ne peut être utilisée telle quelle pour faire du chiffrement ou de la signature, l'étude de sa sécurité n'en demeure néanmoins pas vaine car elle est à la base de nombreux protocoles plus complexes. Un exemple d'une mauvaise utilisation est donné dans le paragraphe 2.5.4.

Comme précisé dans [89], les attaques «mathématiques» contre le RSA peuvent être classées en trois catégories principales⋆ :

1. les attaques exploitant la structure polynomiale du RSA ;
2. les attaques basées sur sa nature multiplicative (homomorphique) ;
3. les attaques résultant d'un mauvais choix de paramètres.

Toutes ces attaques ont été étendues et généralisées aux analogues du RSA construits sur les suites de Lucas et les courbes elliptiques (voir [73] ainsi que [16, 87, 88, 89, 94, 95]). Nous n'allons pas revoir ici ces attaques en détail mais plutôt, à titre d'illustration, montrer comment monter une attaque à messages choisis ([16]) contre les cryptosystèmes LUC [147] et de DEMYTKO [50].

Une des attaques les plus élégantes contre la primitive RSA est due à DAVIDA [48]. Soit $n = pq$ un module RSA et soit $\{e, d\}$ la paire de clés publique/privée correspondante, satisfaisant $e\,d \equiv 1 \pmod{\phi(n)}$. Suivant ce système basique, la signature s d'un message m, vu comme un élément de $\mathbb{Z}/n\mathbb{Z}$, est définie par $s = m^d \bmod n$. La validité de cette signature peut ensuite être vérifiée en testant si $s^e \equiv m \pmod{n}$.

Un attaquant ne connaissant pas la clé privée d peut obtenir la signature s du message m de la façon suivante :

- former un message $\widetilde{m} = m\,r^e \pmod{n}$ où r est un nombre arbitraire ;
- obtenir la signature $\tilde{s}$ de $\widetilde{m}$ (i.e., $\tilde{s} = \widetilde{m}^d \pmod{n}$) ;
- calculer $s = \tilde{s}\,r^{-1} \pmod{n}$.

⋆ Les attaques par fautes et par canaux cachés sont abordées au chapitre 5.

On vérifie aisément que la signature ainsi générée est valide :

$$s^e \equiv (\tilde{s}\, r^{-1})^e \equiv \widetilde{m}\, r^{-e} \equiv m \pmod{n} \ .$$

Cette attaque ne s'étend *a priori* pas aux cryptosystèmes LUC et de DEMYTKO. L'existence d'une falsification à messages choisis nécessitant deux messages a néanmoins été mise en évidence contre LUC dans [15]. Une attaque similaire a été trouvée par KALISKI [108] contre le système de DEMYTKO.

En reformulant l'attaque de DAVIDA avec l'algorithme d'Euclide étendu, il apparaît que des systèmes non-homomorphiques sont également sujet à une attaque à messages choisis ne nécessitant qu'un seul message [16]. Appliquée au RSA, la falsification de la signature du message m s'obtient par les étapes suivantes :

- former un message $\widetilde{m} = m^k \pmod{n}$ où k est un nombre arbitraire, relativement premier avec e ;
- obtenir la signature $\tilde{s}$ de $\widetilde{m}$ (i.e., $\tilde{s} = \widetilde{m}^d \pmod{n}$) ;
- avec l'algorithme d'Euclide étendu, trouver u, v tels que $u\,k + v\,e = 1$ et calculer $s = \tilde{s}^u\, m^v \pmod{n}$.

Ici encore, on vérifie la validité de la signature obtenue :

$$s^e \equiv (\tilde{s}^u\, m^v)^e \equiv \widetilde{m}^u\, m^{ve} \equiv m^{ku+ve} \equiv m \pmod{n} \ .$$

Cette attaque s'applique de la même façon aux cryptosystèmes LUC et de DEMYTKO (voir annexe A, p. 54ff, pour les détails). Passer d'une attaque à deux messages choisis à une attaque à un message choisi peut paraître dérisoire, mais il n'en est rien. Un des avantages avancés des primitives LUC et de DEMYTKO était justement leur caractère non-homomorphique. Cette nouvelle attaque met en évidence que cela n'est pas suffisant pour prémunir ces cryptosystèmes contre des attaques aussi simples et efficaces que celle due à DAVIDA contre la primitive RSA.

Les attaques précédentes ont été décrites dans un contexte de signature électronique, des attaques similaires peuvent être menées lorsque ces primitives de type RSA sont utilisées pour faire du chiffrement. On parle alors d'attaque «des poubelles» ([87] ou annexe A, p. 66ff).

2.2 Réduction de cryptosystèmes

À EUROCRYPT '96, MEYER et MÜLLER [121] ont présenté un nouveau cryptosystème basé sur les courbes elliptiques. Nous montrons ici que ce cryptosystème peut se réduire au cryptosystème de RABIN [137] (voir [90] ou annexe A, p. 80ff). De la même façon, un cryptosystème dû à CHUA et LING [32] peut également se réduire au cryptosystème de RABIN [86].

Nous commençons par une brève description du chiffrement de MEYER-MÜLLER et reportons le lecteur au papier original ([121]) pour plus de détails.

Soit $n = pq$ un module RSA avec $p, q \equiv 11 \pmod{12}$. Le chiffrement d'un message $m \in \mathbb{Z}/n\mathbb{Z}$ est obtenu comme suit :

- former le point $\boldsymbol{P} = (m^2, \lambda m^3)$, avec $\lambda \neq 0$ choisi aléatoirement dans $\mathbb{Z}/n\mathbb{Z}$, sur la courbe
$$E : y^2 = x^3 + a\,x + b \pmod{n}$$
où $a = \lambda^3 \pmod{n}$ et $b = (\lambda^2 - 1)m^6 - a\,m^2 \pmod{n}$;
- calculer $\boldsymbol{Q} = 2\boldsymbol{P}$;
- le texte chiffré correspondant au message m est $c = \big\{a, b, x(\boldsymbol{Q}), \big(\frac{y(\boldsymbol{Q})}{n}\big), \mathrm{lsb}(y(\boldsymbol{Q}))\big\}$.

Il est aisé de retrouver publiquement la valeur de $m^2 \pmod{n}$ à partir du message chiffré $c = \{a, b, x(\boldsymbol{Q}), \big(\frac{y(\boldsymbol{Q})}{n}\big), \mathrm{lsb}(y(\boldsymbol{Q}))\}$. Comme $\boldsymbol{P} = (m^2, \lambda\, m^3) \in E$, il s'ensuit que $(\lambda^2 m^6)^3 \equiv (m^6 + a\,m^2 + b)^3 \pmod{n}$. Par conséquent, en remplaçant λ^3 par a, nous voyons que m^2 est une racine du polynôme $\mathcal{P}_1 \in (\mathbb{Z}/n\mathbb{Z})[X]$,

$$\mathcal{P}_1(X) = (a^2 - 1)X^9 - 3aX^7 - 3bX^6 - 3a^2X^5 - 6abX^4 \\ - (a^3 + 3b^2)X^3 - 3a^2bX^2 - 3ab^2X - b^3 \ .$$

Un second polynôme $\mathcal{P}_2 \in (\mathbb{Z}/n\mathbb{Z})[X]$ dont m^2 est une racine peut être obtenu à partir de la coordonnée en x du point $\boldsymbol{Q}$. Comme $\boldsymbol{Q} = 2\boldsymbol{P}$, nous avons $x(\boldsymbol{Q}) = [(3m^4 + a)/(2\lambda\, m^3)]^2 - 2m^2 \pmod{n}$ et donc nous pouvons définir, après un peu d'algèbre,

$$\mathcal{P}_2(X) = -X^4 + 4\,x(\boldsymbol{Q})X^3 + 2aX^2 + [8b + 4a\,x(\boldsymbol{Q})]X - a^2 + 4b\,x(\boldsymbol{Q}) \ .$$

Étant donné que m^2 est une racine des polynômes $\mathcal{P}_1$ et $\mathcal{P}_2$, m^2 est également une racine de leur plus grand commun diviseur, $\mathcal{R} = \mathrm{pgcd}(\mathcal{P}_1, \mathcal{P}_2)$. Le polynôme $\mathcal{R}$ est de degré 1 avec une très forte probabilité [39]. La résolution de ce polynôme en X fournit donc la valeur de $m^2 \pmod{n}$ [90].

Par conséquent, le cryptosystème de Meyer et de Müller n'apporte rien par rapport au cryptosystème, plus simple, de Rabin. En particulier, la connaissance de deux messages «stéréotypés» (au sens de [39]) chiffrés suffit à retrouver les messages clairs correspondants, et non onze messages comme annoncé dans [121].

2.3 Analyse du standard PKCS #1

Les documents PKCS (Public Key Cryptographic Standards), maintenus par les laboratoires RSA, sont des spécifications ayant pour but le déploiement rapide de la cryptographie à clé publique. Publiés pour la première fois en 1991, ces documents ont été adoptés dans de nombreux produits et sont repris dans plusieurs standards. Parmi ceux-ci, PKCS #1 [141] décrit des techniques d'encodage (paddings) pour chiffrer ou signer des messages avec la primitive RSA.

Soit n un module RSA et soit e l'exposant public de chiffrement. Dans ce qui suit, la longueur en octets d'un nombre x (vu comme une chaîne d'octets) sera notée $|x|$. Avec la version 1.5 de PKCS #1, un message m de $|m|$ octets avec $|m| \leq |n| - 11$ est d'abord transformé en un message «paddé»

$$\text{PKCS1\_v1.5}(m, r') = \mathtt{0002}_{16} \| r' \| \mathtt{00}_{16} \| m$$

où r' est un nombre aléatoire dont tous les octets sont non nuls et tel que $|r'| = |n| - 3 - |m| \geq 8$. Ce message paddé est ensuite mis à la puissance e pour former le texte chiffré correspondant $c = \{\text{PKCS1\_v1.5}(m, r')\}^e \bmod n$. En posant $r = (\mathtt{02}_{16} \| r')$ et $\beta = 8(|m| + 1)$, nous pouvons ré-écrire le message chiffré comme

$$c = (r 2^\beta + m)^e \bmod n \ .$$

Des techniques cryptanalytiques avancées, basées sur l'algorithme de réduction LLL, ont été mises au point par Coppersmith [38]. Une de celles-ci est résumée dans la proposition suivante.

Proposition 1 (Coppersmith). *Soit $\mathcal{P}(x)$ un polynôme de degré δ en une variable, défini modulo un entier n dont la factorisation est inconnue et soit X une borne sur la solution désirée x_0. Si $X \leq n^{1/\delta}$ alors toutes les solutions telles que $\mathcal{P}(x_0) = 0 \pmod n$ et $-X \leq x_0 \leq X$ peuvent être trouvées en un temps polynômial en $(\log n, 2^\delta)$.* □

Ainsi, appliquée au RSA avec un petit exposant ($e = 3$), cette proposition permet de passer à travers le padding PKCS #1 v1.5 dès qu'un attaquant connaît les deux tiers de $r\,2^\beta + m$. En particulier, cette attaque ne s'applique pas lorsque les messages à chiffrer sont courts (par exemple, une clé DES ou AES).

Une autre attaque contre le padding PKCS #1 v1.5 utilisé avec un petit exposant a été présentée dans [44] (voir également annexe A, p. 84ff). Cette attaque suppose que les Z derniers bits du message clair m sont tous égaux à zéro. Un exemple concret est lorsque, pour des raisons d'efficacité, un utilisateur utilise RSA-PKCS #1 v1.5 pour chiffrer des messages courts en complétant ces messages par un nombre approprié d'octets nuls de sorte que seulement 8 octets (i.e., le minimum avec PKCS #1 v1.5) doivent être tirés aléatoirement pour former r'. Dans ce cas, en notant $m = \bar{m} 2^Z$, nous obtenons $c = \{2^Z (r 2^{\beta - Z} + \bar{m})\}^e \bmod n$. Par conséquent, si un même message m est chiffré deux fois (i.e., $c_i = \{2^Z (r_i 2^{\beta - Z} + \bar{m})\}^e \bmod n$ pour $i = 1, 2$), un attaquant peut évaluer

$$\begin{aligned} \Delta \ &:= \frac{c_1 - c_2}{2^{eZ}\, 2^{\beta - Z}} \bmod n \\ &\equiv \underbrace{(r_1 - r_2)}_{:=\omega} \underbrace{\Big[\sum_{j=0}^{e-1} (r_1\, 2^{\beta - Z} + \bar{m})^{e-1-j} (r_2\, 2^{\beta - Z} + \bar{m})^j \Big]}_{:=v} \pmod n \ . \end{aligned}$$

Supposant que $r_1 > r_2$ et que le nombre Z de bits nuls est suffisamment élevé pour que $0 < \omega v < n$, alors la relation précédente est valable dans $\mathbb{Z}$ et donc

$\omega = r_1 - r_2$ est un diviseur de Δ. Par conséquent, en extrayant les petits facteurs de Δ, un attaquant peut construire un candidat pour ω et de là retrouver le message clair :

- pour chaque diviseur Δ_j de Δ, calculer, dans $(\mathbb{Z}/n\mathbb{Z})[X]$,

$$\mathcal{R}_j(X) = \text{pgcd}(X^e - c_1, (X - 2^\beta \Delta_j)^e - c_2)\,;$$

- si $\Delta_j = \omega$ alors, avec une très forte probabilité, $\mathcal{R}_j(X) = X - \text{PKCS1\_V1.5}(m, r'_1) \pmod n$, dont la résolution fournit la valeur de m.

Cette attaque est effective (i.e., la condition $\omega v < n$ est vérifiée) dès que $Z > (e-1)R + (e-2)M + 10e - 34$ où R et M désignent respectivement les tailles en bits de r' et de $\bar{m}$ [44]. Par contre, l'attaque dérivée de la proposition 1 n'est effective que si $Z > (e-1)R + (e-1)M - 24$.

Par ailleurs, il est possible de générer des messages chiffrés conformes à PKCS #1 v1.5 sans connaître les messages clairs correspondants, avec une certaine probabilité. Cette propriété est exploitée par Bleichenbacher pour monter une attaque à messages chiffrés choisis [14]. Certaines contre-mesures sont présentées dans [17]. Cette dernière attaque ainsi que l'attaque précédente ont fait évoluer le standard PKCS #1 qui en est maintenant à la version v2.1 (juin 2002) [141]. La version v1.5 n'est supportée que pour des raisons de compatibilité.

PKCS #1 spécifie également un format de signature pour la primitive RSA. Une des attaques les plus sophistiquées contres les signatures RSA est due à Coron, Naccache et Stern [46] (voir également [40, 66]). Indépendamment, ce type d'attaque a ensuite été étendu aux systèmes de type Rabin ([92] ou annexe A, p. 97ff). Contrairement au RSA, l'adversaire retrouve les clés privées et peut par conséquent falsifier la signature d'un message de son choix. De plus, l'utilisation d'exposants publics (pairs) plus grands ne réduit pas la complexité de l'attaque. Toutes ces attaques sont cependant de nature théorique contre le padding de signature PKCS #1 v1.5. Il est néanmoins recommandé d'utiliser la version v2.1 (PSS) [141].

2.4 Notions de sécurité prouvée

La sécurité prouvée s'est largement développée ces quinze dernières années comme un moyen de valider diverses constructions cryptographiques. Tout comme la vitesse d'exécution, les ressources mémoire nécessaires ou la bande passante utilisée, la sécurité prouvée est un attribut qui permet de juger de la «qualité» d'un cryptosystème. Il ne faut cependant pas perdre de vue qu'une preuve de sécurité peut être incomplète voire invalide ou que le modèle de sécurité considéré peut être inadéquat ou limité. L'exemple le plus marquant est certainement la mise en évidence par Shoup [146] que la preuve de sécurité pour OAEP donnée dans [6] était incorrecte (voir cependant [61] pour son application au RSA). D'autres exemples concrets dans le cadre des schémas de signature sont discutés dans [148].

Dans cette section, nous étudions la sécurité prouvée pour les schémas de chiffrement. La notion de sécurité habituellement retenue pour les schémas de chiffrement à clé publique est celle de sécurité contre les attaques à messages chiffrés choisis [138] (appelée IND-CCA2 dans [5]). Cette notion est définie par le scénario d'attaque suivant.

Étape 1 : L'algorithme de génération de clés pour le cryptosystème est lancé, produisant une paire de clés publique/privée. L'adversaire obtient une copie de la clé publique mais pas de la clé privée.

Étape 2 : L'adversaire exécute une série de requêtes arbitraires à un oracle de déchiffrement. Chaque requête est un message chiffré c, lequel est déchiffré par l'oracle de déchiffrement, en utilisant la clé privée. Le résultat de chaque déchiffrement est donné à l'adversaire.

Étape 3 : L'adversaire choisit ensuite deux messages clairs de même longueur, m_0 et m_1, et les soumet à un oracle de chiffrement. Cet oracle de chiffrement choisit aléatoirement $b \in \{0,1\}$, chiffre m_b et renvoie le chiffré correspondant c^* à l'adversaire.

Étape 4 : L'adversaire peut exécuter une deuxième série de requêtes à l'oracle de déchiffrement pour des messages chiffrés arbitraires $c \neq c^*$.

Étape 5 : L'adversaire retourne $\tilde{b} \in \{0,1\}$ s'il estime que c^* correspond au chiffrement de $m_{\tilde{b}}$. L'attaque est fructueuse si $\tilde{b} = b$.

L'avantage de l'adversaire dans ce scénario d'attaque est défini par $|\Pr[\tilde{b} = b] - 1/2|$. Brièvement, un cryptosystème est dit sûr contre les attaques à messages chiffrés choisis si l'avantage de tout adversaire est négligeable.

La notion de «validité» pour les messages chiffrés est laissée vague (ou du moins n'est pas explicitement abordée) dans le scénario d'attaque précédent. En particulier, dans les étapes 2 et 4, on pourrait supposer que l'adversaire n'effectue des requêtes qu'avec des messages chiffrés valides, c.-à-d. qui sont le résultat du chiffrement de messages pris dans l'ensemble de définition des messages clairs.

Même si l'adversaire n'a habituellement aucun avantage à sonder l'oracle de déchiffrement avec des messages invalides, certaines attaques ont été reportées contre des primitives de chiffrement ([62, 70]). De façon peut-être plus surprenante, le schéma de chiffrement à sécurité prouvée EPOC [129], basé sur la primitive d'Okamoto-Uchiyama [128], est également sujet à une telle attaque [98] (ou annexe A, p. 112ff).

Les systèmes EPOC existent en deux versions : EPOC-1 et EPOC-2. Soient $n = p^2 q$ avec p et q des premiers de k bits, $g_p = g^{p-1} \bmod p^2$ d'ordre p et $h = {h_0}^n \bmod n$ où $g, h_0 \in (\mathbb{Z}/n\mathbb{Z})^*$. Soient également p_{Len}, m_{Len} et r_{Len} tels que $p_{\text{Len}} = k$ et $m_{\text{Len}} + r_{\text{Len}} \leq p_{\text{Len}} - 1$ et une fonction de hachage H. Dans EPOC-1, le chiffrement et déchiffrement d'un message s'effectuent comme suit :

– Un message $m \in \{0,1\}^{m_{\text{Len}}}$ est chiffré en

$$c = g^{(m\|r)} h^{H(m\|r)} \bmod n$$

où r est choisi de façon uniforme dans $\{0,1\}^{r_{\text{Len}}}$.

– Étant donné le chiffré c, on calcule d'abord

$$X = \frac{L(c^{p-1} \bmod p^2)}{L(g_p)} \bmod p$$

où $L(x) = (x-1)/p$. Si $g^X h^{H(X)} \bmod n = c$ alors le message clair correspondant est $[X]^{m_{\text{Len}}}$ (i.e., les m_{Len} bits de poids fort de X vu comme une chaîne de $(m_{\text{Len}} + r_{\text{Len}})$ bits) ; sinon $\perp$ est retourné, indiquant que le message chiffré c est invalide.

Une des particularités de la primitive d'Okamoto-Uchiyama est que l'ensemble complet des «messages» valides, $[0, p[$, est tenu secret. Pour construire un cryptosystème à partir de cette primitive, il faut travailler sur un sous-ensemble $[0, T[$ avec $T < p$. Dans EPOC-1, nous avons $T = 2^{p_{\text{Len}}-1}$. Ainsi, deux catégories de messages chiffrés invalides peuvent être distingués :

– les «messages» $X = (m\|r)$ tels que $X \geq p$; et
– les «messages» $X = (m\|r)$ tels que $T \leq X < p$.

Seuls les messages de la première catégorie donneront lieu au symbole $\perp$ lors du déchiffrement. Cette dissymétrie peut être exploitée par un adversaire pour retrouver bit à bit la valeur du secret p. Comme p est un nombre de p_{Len} bits, l'adversaire sait que p se trouve dans l'intervalle $I_0 =]2^{p_{\text{Len}}-1}, 2^{p_{\text{Len}}}[$. Il choisit donc un message m tel que $X = (m\|r) \in I_0$ et calcule le texte chiffré correspondant avec l'algorithme de chiffrement. Si l'algorithme de déchiffrement retourne $\perp$, alors l'adversaire déduit que $p \leq X$; sinon, il déduit que $p > X$. L'adversaire réitère ensuite l'attaque avec l'intervalle $I_1 =]2^{p_{\text{Len}}-1}, X]$ ou $I_1 =]X, 2^{p_{\text{Len}}}[$ et ainsi de suite jusqu'à ce que l'intervalle obtenu devienne suffisamment petit pour retrouver la valeur de p. Il est cependant facile de contrer cette attaque : il suffit de vérifier explicitement lors du déchiffrement la condition $X < 2^{p_{\text{Len}}-1}$ et de retourner $\perp$ si celle-ci n'est pas vérifiée.

Des attaques similaires peuvent être menées contre EPOC-2 ou contre le protocole PPTK de Fischlin [57] (voir [98] ou annexe A, p. 112ff). La modification proposée pour EPOC-1 peut néanmoins être adaptée pour rendre ces attaques inapplicables.

Les attaques précédentes mettent en évidence l'importance de l'aspect définitionnel de la cryptographie. Elles mettent également en évidence qu'une preuve de sécurité n'est pas absolue et qu'elle peut être invalidée dans des situations non prévues par le modèle de sécurité.

Une autre extension du modèle de sécurité pour les schémas de chiffrement est considérée dans [109]. Dans ce modèle étendu, en plus d'avoir accès à un oracle de déchiffrement, un adversaire peut effectuer un «vidage de mémoire» de l'oracle de déchiffrement à tout moment ; la seule restriction étant que les données secrètes (par exemple, les clés) sont inaccessibles : en particulier, les opérations manipulant ces données secrètes sont effectuées de façon atomique. Dans [109], il est montré que la plupart des cryptosystèmes, y compris le très populaire RSA-OAEP [6, 141], est sujet à ce type d'attaques étendues. Toutefois, deux exceptions sont à noter : le cryptosystème de Tsiounis et Yung [152] et le cryptosystème

de Schnorr et Jakobsson [142]. Ces deux systèmes présentent la particularité que la validité des messages chiffrés est testée à partir des paramètres publics uniquement, avant l'opération de déchiffrement proprement dite. Dans une moindre mesure, le cryptosystème de Cramer et Shoup [47] résiste également à ces attaques étendues : bien que la validation se fasse avant l'opération de déchiffrement, elle requiert néanmoins la connaissance de paramètres secrets.

2.5 Autres attaques

2.5.1 Paradoxes de sécurité

Paradoxalement, ce qui semble être une amélioration universelle pour augmenter la sécurité ou la fiabilité d'un cryptosystème sur le plan théorique peut introduire de nouvelles failles sur le plan pratique [96, 97]. Par exemple, la primitive RSA peut s'avérer plus sûre si

1. elle est implantée sans détection de fautes (par exemple, avec la contre-mesure décrite dans [145]) ;
2. elle est utilisée sans padding prouvé sûr contre les attaques à messages chiffrés choisis, tel OAEP [6, 141] ;
3. elle est utilisée avec des modules RSA dont la factorisation est supposée plus facile [144].

Ces trois paradoxes de sécurité proviennent du fait que les améliorations introduites engendrent davantage de calculs ou des calculs différents ; lesquels peuvent être, dans certains contextes d'utilisation ou environnements de travail, observés et exploités par un adversaire.

2.5.2 Partage de clés

Un protocole efficace pour générer un jeu de paramètres RSA par deux parties de sorte qu'aucune des parties ne connaisse la factorisation du module RSA mais leur permet de déchiffrer conjointement un texte chiffré a été proposé par Cocks [37]. Malheureusement, il est possible pour une partie malhonnête de dévier du protocole et de retrouver la factorisation du module RSA [84] (voir aussi [13]). Des méthodes alternatives de génération de paramètres RSA partagés peuvent être trouvées dans [21, 58, 133, 63].

Un protocole d'échange de clés dû à Harn et Lin [67] est cryptanalysé dans [156]. Les failles trouvées dans ce protocole ainsi que dans le protocole de Cocks ([37]) illustrent une fois de plus l'importance des preuves de sécurité en cryptographie.

2.5.3 Signatures de groupe

Un schéma de signature de groupe** permet aux membres d'un groupe de signer de façon anonyme au nom du groupe. Cependant, dans des cas exceptionnels, l'anonymat du signeur peut être levé par une autorité de groupe. Une des

** Ces schémas sont discutés plus en détail à la section 3.1.

tâches les plus difficiles lors de l'élaboration d'un schéma de signature de groupe est d'empêcher les attaques par collusion : des membres du groupe complices produisent ensemble des signatures intraçables par l'autorité de groupe. Par exemple, les schémas de CAMENISCH et STADLER [26], de LEE et CHANG [114] et de TSENG et JAN [151] ne vérifient pas cette propriété [3]. Par ailleurs, il est respectivement montré dans [78, 77] que les deux derniers schémas ([114, 151]), en plus d'être sujet aux attaques par collusion, sont également universellement falsifiables.

2.5.4 Micro-paiements

Soit $n = p_1 p_2$ un module de 1024 bits de type RSA tel que q, un premier de 160 bits, divise $p_1 - 1$. La sécurité du système de micro-paiement MICROCAST [54] repose sur la difficulté de trouver une solution (A, p) à l'équation

$$\mathcal{P} A^x \equiv g^p \pmod{n}$$

étant donné $\mathcal{P}, g, x, n$ et où g est un élément d'ordre q.

Comme p_2 est choisi de façon aléatoire, q ne divise pas $p_2 - 1$ avec une très forte probabilité. De plus, comme $g^q \equiv 1 \pmod{\{p_1, p_2\}}$, nous déduisons que $g \equiv 1 \pmod{p_2}$ et donc que $g \not\equiv 1 \pmod{p_1}$, ce qui implique $\text{pgcd}(g-1, n) = p_2$ [75].

Une fois la factorisation de n connue, il devient trivial de résoudre l'équation précédente en calculant, pour un p arbitraire,

$$A = \left[\frac{g^p}{\mathcal{P}}\right]^{x^{-1} \bmod (p_1-1)(p_2-1)} \bmod n \ .$$

Par ailleurs, même si p_2 est construit de sorte que q divise $p_2 - 1$, il est encore possible de trouver une solution (A, p) à l'équation car MICROCAST suppose que $\mathcal{P}$ est de la forme $\mathcal{P} = g^\alpha \bmod n$. Par conséquent, $\mathcal{P}$ et donc $A \in \langle g \rangle$. De plus, comme q divise $p_1 - 1$ et $p_2 - 1$, il s'ensuit que q divise $p_1 p_2 - 1 = n - 1$. Dans ce cas, une solution au problème est donnée par $\left((g^p/\mathcal{P})^{x^{-1} \bmod (n-1)} \bmod n, p\right)$ pour un p arbitraire [75].

Chapitre 3

Nouveaux schémas cryptographiques

3.1 Signature de groupe

Le concept de signature de groupe (à ne pas confondre avec le concept de multi-signature) a été introduit par Chaum et van Heyst [29]. Les schémas de signature de groupe permettent aux membres d'un groupe de signer des messages au nom du groupe de façon anonyme pour tous sauf pour une autorité de groupe, laquelle est capable de retrouver l'identité du signeur. De tels schémas trouvent de nombreuses applications, notamment dans les domaines de la monnaie électronique, des systèmes d'enchères ou de vote électronique [30, 118, 125, 150].

Les schémas de signature de groupe les plus récents sont basés sur le même principe : un premier schéma de signature est utilisé pour émettre un certificat d'appartenance au groupe et un deuxième schéma de signature est utilisé pour produire la signature de groupe elle-même en prouvant la possession de ce certificat. Comme illustré au paragraphe 2.5.3, une des difficultés majeures dans la construction d'un schéma de signature de groupe est de prouver la résistance contre les attaques par collusion. Cette propriété est assurée par le premier schéma de signature. En effet, chaque membre du groupe reçoit un certificat qui n'est autre qu'une signature par l'autorité de groupe d'un message aléatoire choisi par le membre. La collusion de plusieurs membres du groupe peut ainsi être vue comme un adversaire unique menant une attaque à messages choisis.

En d'autres termes, toute la difficulté consiste à construire un premier schéma de signature (utilisé pour émettre les certificats) sûr contre les attaques à messages choisis de sorte que le deuxième schéma de signature (utilisé pour produire les signatures de groupe) soit efficace. La sécurité du premier schéma de signature peut cependant être limitée aux attaques passives (i.e., à messages connus) si les messages aléatoires utilisés pour produire les certificats d'appartenance au groupe sont choisis conjointement pour l'utilisateur voulant rejoindre le groupe et l'autorité de groupe.

L'état de l'art pour les schémas de signature de groupe est donné par le schéma ACJT [2] (ou annexe A, p. 127ff). Il faut toutefois noter le très élégant schéma basé sur les cartes à puces, dû à Canard et Girault [27].

Dans le schéma ACJT, l'autorité de groupe construit un module RSA $n = pq$ où $p = 2p' + 1$ et $q = 2q' + 1$ avec p', q' premiers. Il publie également deux éléments $a, a_0 \in \mathrm{QR}(n)$. Un certificat d'appartenance au groupe, correspondant à la clé publique $B_i = a^{x_i} \bmod n$ où x_i est un paramètre privé de l'utilisateur i,

est donné par une paire (A_i, e_i) telle que

$$A_i{}^{e_i} \equiv B_i\, a_0 \equiv a^{x_i} a_0 \pmod{n} \quad \text{avec } x_i \in \mathcal{X},\ e_i \text{ premier} \in \mathcal{E},$$

et $\mathcal{X} \subset \mathcal{E} \subset [0, n[$.

Nous donnons ci-dessous une version simplifiée de la preuve apparaissant dans [2] de la résistance contre les attaques par collusion.

Proposition 1. *Avec les notations précédentes, sous l'hypothèse RSA forte, il n'existe aucun adversaire qui, ayant pour entrée n, a, a_0 et K triplets (A_1, e_1, x_1), $\ldots, (A_K, e_K, x_K)$, peut produire avec une probabilité non négligeable un autre triplet $(\hat{A}, \hat{e}, \hat{x})$ tel que $\hat{A} = (a^{\hat{x}} a_0)^{1/\hat{e}} \bmod n$ avec $\hat{x} \in \mathcal{X}$ et $\hat{e} \in \mathcal{E}$.*[***]

Démonstration. La preuve est par l'absurde. Supposons qu'il existe un attaquant $\mathcal{M}$ qui contredit l'énoncé de la proposition. Nous allons montrer que cet attaquant $\mathcal{M}$ peut être utilisé pour résoudre une instance du problème RSA fort, à savoir trouver une solution $(u, e) \in \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}_{>1}$ telle que $u^e \equiv z \pmod{n}$, étant donné $n = (2p'+1)(2q'+1)$ et $z \in \mathrm{QR}(n)$.

Jeu 1

1. Choisir $x_1, \ldots, x_K \in_R \mathcal{X}$ et $e_1, \ldots, e_K$ premiers $\in_R \mathcal{E}$.
2. Poser $a := z^{\prod_{1 \le l \le K} e_l} \bmod n$.
3. Choisir $r' \in_R [1, n^2]$ et poser $r := r' + \mathfrak{e}$ et $a_0 := a^r \bmod n$, où $\mathfrak{e} := \max_{e \in \mathcal{E}} e$.
4. Pour tout $1 \le i \le K$, calculer $A_i = z^{(x_i + r) \prod_{1 \le l \le K; l \ne i} e_l} \bmod n$.
5. Lancer $\mathcal{M}$ avec en entrée n, a, a_0 et $(A_1, e_1, x_1), \ldots, (A_K, e_K, x_K)$ et obtenir en sortie $(\hat{A}, \hat{e}, \hat{x})$ tel que $\hat{x} \in \mathcal{X}$, $\hat{e} \in \mathcal{E}$, et $\hat{A}^{\hat{e}} = a^{\hat{x}} a_0 \bmod n$.
6. Si $\mathrm{pgcd}(\hat{e}, e_j) \ne 1$ pour un certain $1 \le j \le K$ alors retourner $\perp$ et quitter. Sinon, poser $\tilde{e} := (\hat{x} + r) \prod_{1 \le l \le K} e_l$. Étant donné que $\mathrm{pgcd}(\hat{e}, e_j) = 1$ pour tout $1 \le j \le K$, il s'ensuit que $d := \mathrm{pgcd}(\hat{e}, \tilde{e}) = \mathrm{pgcd}(\hat{e}, (\hat{x} + r)) \ne \hat{e}$ avec une probabilité $\ge 1 - \frac{1}{\mathfrak{e}}$. En effet, $\hat{e} < \hat{x} + r$ et donc $\mathrm{pgcd}(\hat{e}, (\hat{x} + r)) = \hat{e}$, ce qui est équivalent à

$$\hat{e} \mid (\hat{x} + r) \iff (\hat{x} + r_1) + r_2\, p'q' \equiv 0 \pmod{\hat{e}} \qquad (\dagger)$$

avec $r_1 := r \bmod p'q'$ et $r_2 = (r - r_1)/(p'q')$. La valeur r_1 peut être vue comme l'information «maximale» que $\mathcal{M}$ peut obtenir à propos de r : r intervient modulo $p'q'$ dans l'expression de a_0 et de $A_1, \ldots, A_K$. De plus, nous pouvons supposer que $\mathrm{pgcd}(\hat{e}, p'q') = 1$ (sinon n pourrait être factorisé). Comme $r \gg \hat{e}$, la distribution de $r \bmod \hat{e}$ est proche de la distribution uniforme sur $[0, \hat{e}[$ et donc la probabilité que r_2 soit solution de $(\dagger)$ est au plus $\frac{1}{\mathfrak{e}}$.

Avec l'algorithme d'Euclide étendu, nous trouvons $\alpha, \beta \in \mathbb{Z}$ tels que $\alpha\, \hat{e} + \beta\, \tilde{e} = d$. Comme $\hat{A}^{\hat{e}} \equiv z^{\tilde{e}} \pmod{n}$, il est facile de voir que $u := z^{\alpha}\, \hat{A}^{\beta} \bmod n$ et $e := \hat{e}/d$ est une solution de notre problème initial.

[***] Il est à noter que $\hat{e}$ n'est pas supposé être premier car, pour des raisons d'efficacité, cette propriété n'est pas explicitement vérifiée dans les signatures de groupe ACJT [2].

Le jeu suivant retourne une solution lorsque $\text{pgcd}(\hat{e}, e_j) \neq 1$ pour un certain $1 \leq j \leq K$. Il est à noter que la condition $\text{pgcd}(\hat{e}, e_j) \neq 1$ signifie $\text{pgcd}(\hat{e}, e_j) = e_j$ car e_j est premier, laquelle implique à son tour $\hat{e} = e_j$ par le choix de $\mathcal{E}$.

Jeu 2

1. Choisir $x_1, \ldots, x_K \in \mathcal{X}$ et $e_1, \ldots, e_K$ premiers $\in \mathcal{E}$.
2. Choisir $j \in_R \{1, \ldots, K\}$ et poser $a := z^{\prod_{1 \leq l \leq K; l \neq j} e_l} \bmod n$.
3. Choisir $r \in_R \mathcal{X}$ et poser $A_j := a^r \bmod n$ et $a_0 := A_j^{e_j} / a^{x_j} \bmod n$.
4. Pour tout $1 \leq i \leq K$, $i \neq j$, calculer $A_i = z^{(x_i + e_j r - x_j) \prod_{1 \leq l \leq K; l \neq i,j} e_l} \bmod n$.
5. Lancer $\mathcal{M}$ avec en entrée n, a, a_0 et obtenir en sortie $(A_1, e_1, x_1), \ldots, (A_K, e_K, x_K)$ tels que $(\hat{A}, \hat{e}, \hat{x})$ such that $\hat{x} \in \mathcal{X}$, $\hat{e} \in \mathcal{E}$, and $\hat{A}^{\hat{e}} = a^{\hat{x}} a_0 \bmod n$.
6. Si $\hat{e} \neq e_j$ alors retourner $\perp$ et quitter. Sinon, nous avons $\hat{e} = e_j$ et par conséquent $\hat{x} \neq x_j$. En posant $\tilde{e} := (\hat{x} - x_j) \prod_{1 \leq l \leq K; l \neq j} e_l$, nous avons $(\hat{A}/A_j)^{e_j} \equiv a^{\hat{x} - x_j} \equiv z^{\tilde{e}} \pmod{n}$. Par l'algorithme d'Euclide étendu, nous obtenons $\alpha, \beta \in \mathbb{Z}$ tels que $d := \text{pgcd}(e_j, \tilde{e}) = \alpha\, e_j + \beta\, \tilde{e} = \text{pgcd}(e_j, \hat{x} - x_j) \neq e_j$ par le choix de $\mathcal{E}$ et de $\mathcal{X}$. Ainsi, $u := z^\alpha\, (\hat{A}/A_j)^\beta \bmod n$ et $e := e_j/d$ est une solution à notre problème.

En jouant aléatoirement les jeux 1 et 2, une instance du problème RSA fort pourrait être résolu, ce qui contredit les hypothèses. Par conséquent, l'attaquant $\mathcal{M}$ n'existe pas. □

La preuve de la sécurité des autres propriétés du schéma ACJT (i.e., non-falsification, anonymat, traçabilité, non-repudiation) est classique (voir [2] ou annexe A, p. 127ff).

3.2 Conversion générique IND-CCA2

Le niveau de sécurité standard pour les schémas de chiffrement à clé publique est la sécurité IND-CCA2, i.e., «indistinguabilité» (IND) contre les attaques adaptatives à messages chiffrés choisis (CCA2) [138] (voir section 2.4). De façon générale, un niveau de sécurité est défini en couplant un but de sécurité avec un modèle d'attaque [5]. Un but de sécurité plus modeste que l'indistinguabilité (et en un sens minimal) est le caractère «à sens unique» (OW) : un adversaire ne peut pas retrouver complètement le message clair à partir d'un message chiffré donné. Par ailleurs, en notant qu'en cryptographie à clé publique il est toujours possible pour un attaquant de produire des messages chiffrés de son choix (CPA), un niveau de sécurité minimal pour un schéma de chiffrement à clé publique est la sécurité OW-CPA. D'autres buts de sécurité ont été définis tels que la non-malléabilité (NM) [53] ainsi que d'autres modèles d'attaques tels que les attaques non-adaptatives à messages chiffrés choisis (CCA1) [124] ou les attaques par vérification des messages clairs (PCA) [126]. Dans le scénario d'attaque PCA, l'adversaire a accès un oracle qui étant donné une paire (m, c) vérifie si oui ou

non le message clair m correspond au message chiffré c (i.e., si c est le chiffré de m).

Les conversions génériques de schémas de chiffrement à clé publique faibles en schémas plus forts ont été étudiées par FUJISAKI, OKAMOTO et POINTCHEVAL dans [59, 60, 132]. Ces conversions présentent la particularité qu'un «rechiffrement» doit être effectué lors du déchiffrement pour assurer la sécurité IND-CCA2. Plus récemment, OKAMOTO et POINTCHEVAL ont imaginé la conversion REACT [127]. Soient G et H des fonctions de hachage et soit u un nombre aléatoire. Un message m est chiffré en

$$\text{REACT}(m) = \underbrace{\mathcal{E}_{pk}(R,u)}_{=c_1} \,\|\, \underbrace{\mathtt{E}_{G(R)}(m)}_{=c_2} \,\|\, \underbrace{H(R,m,c_1,c_2)}_{=c_3}$$

- pour chaque message à chiffrer, une clé de session aléatoire R est générée et chiffrée par un schéma à clé publique $\mathcal{E}_{pk}$: $c_1 = \mathcal{E}_{pk}(R,u)$;
- ensuite, la clé de session R est utilisée pour chiffrer le message m par un schéma à clé secrète $\mathtt{E}_K$: $c_2 = \mathtt{E}_K(m)$ avec $K = G(R)$;
- enfin, un «checksum» est annexé à cette construction : $c_3 = H(R,m,c_1,c_2)$.

Contrairement aux conversions précédentes, REACT ne nécessite pas de rechiffrement lors du déchiffrement pour garantir la sécurité IND-CCA2, dans le modèle de l'oracle aléatoire. De plus, le chiffrement hybride REACT ne requiert que des propriétés très faibles pour les schémas sous-jacents : $\mathcal{E}_{pk}$ doit être OW-PCA et $\mathtt{E}_K$ doit être sémantiquement sûr contre les attaques passives (par exemple un simple XOR).

Dans certaines applications, la taille des textes chiffrés peut être critique pour des raisons de bande passante ou de stockage. Nous présentons ci-dessous une variante de la conversion précédente, appelée GEM (de l'anglais «Generic Encryption Method») [42] (ou annexe A, p. 143ff), qui assure le même niveau

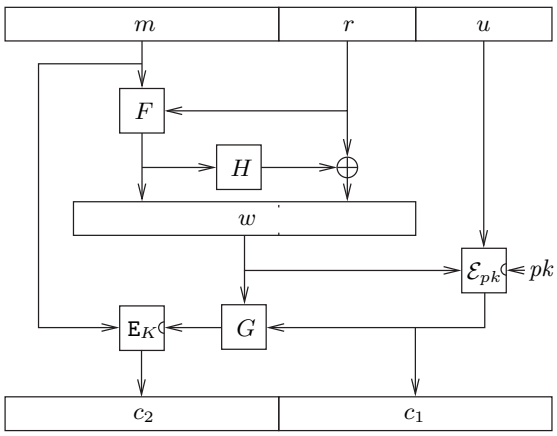


Fig. 3.1. Conversion générique GEM.

de sécurité que REACT à partir des mêmes primitives, mais sans checksum additionnel.

Soient F, G et H des fonctions de hachage et soient r et u des nombres aléatoires. Le chiffrement du message m est donné par

$$\text{GEM}(m) = \underbrace{\mathcal{E}_{pk}(w,u)}_{=c_1} \,\|\, \underbrace{\mathtt{E}_K(m)}_{=c_2} \quad \text{où} \begin{cases} s = F(m,r), w = s \,\|\, r \oplus H(s) \\ K = G(w, c_1) \end{cases} .$$

La conversion REACT présente l'avantage de pouvoir opérer «à la volée» : la composante à clé publique (i.e., c_1) ne dépend pas du message à chiffrer et peut donc être pré-calculée. Ceci n'est pas le cas de la conversion GEM ; laquelle peut être vue comme la meilleure alternative à REACT lorsque le chiffrement à clé publique sous-jacent est rapide (comme le RSA avec un petit exposant public) ou lorsque les gains en bande passante et/ou en mémoire sont une priorité.

Les conversions REACT et GEM ont ensuite été étendues à des messages de taille arbitraire [43] (ou annexe A, p. 158ff). Telles que présentées, ces conversions sont cependant sujet à une attaque (théorique) [71] dans un modèle de sécurité étendu. Dans cette attaque, les challenges m_0 et m_1 (voir étape 3 du scénario IND-CCA2, page 8) sont vus comme des vecteurs qui sont choisis composante par composante, de manière adaptative (dans le modèle de sécurité classique, les challenges m_0 et m_1 sont vus de façon atomique).

3.3 Padding universel

La primitive RSA ne peut être utilisée telle quelle pour chiffrer ou signer des messages (voir section 2.1). Une fonction de padding doit être appliquée avant l'exponentiation modulaire. Comme les buts de sécurité sont différents pour le chiffrement et la signature, deux paddings sont proposés : OAEP [6, 141] est recommandé pour le chiffrement RSA et PSS [7, 141] pour la signature.

Il existe deux versions du padding PSS : avec ou sans recouvrement de message. Dans sa version avec recouvrement de message, la signature d'un message avec RSA-PSS s'effectue comme suit. Soit n un module RSA de k bits et soit d l'exposant privé de signature. Soient également $G : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1}$ et $H : \{0,1\}^{k-k_1} \to \{0,1\}^{k_1}$ des fonctions de hachage. La signature d'un message m s'effectue comme suit :

– Un message $m \in \{0,1\}^{k-k_0-k_1}$ est paddé en

$$\text{PSS-R}(m,r) = \omega \,\|\, \sigma \quad \text{avec} \begin{cases} \omega = H(m\|r) \\ \sigma = G(\omega) \oplus (m\|r) \end{cases}$$

où r est choisi de façon uniforme dans $\{0,1\}^{k_0}$. La signature du message m est alors donnée par

$$s = \text{PSS-R}(m,r)^d \bmod n \ .$$

– Étant la signature s, on calcule d'abord

$$\omega \,\|\, \sigma = s^e \bmod n \quad \text{et} \quad m \,\|\, r = G(\omega) \oplus \sigma$$

où e est l'exposant public de vérification. Si $H(m\|r) = \omega$, alors s est une signature valide du message m.

De façon surprenante, le padding PSS peut également être utilisé pour chiffrer des messages [45] (ou annexe A, p. 170ff). Avec les notations précédentes, e et d représentent respectivement les exposants public de chiffrement et secret de déchiffrement. Le chiffrement d'un message m est donné par :

– Un message $m \in \{0,1\}^{k-k_0-k_1}$ est chiffré en

$$c = \text{PSS-R}(m, r)^e \bmod n$$

avec $r \in_R \{0,1\}^{k_0}$.

– Étant le chiffré c, on calcule d'abord

$$\omega \,\|\, \sigma = c^d \bmod n \quad \text{et} \quad m \,\|\, r = G(\omega) \oplus \sigma \ .$$

Si $H(m\|r) = \omega$ alors le message clair correspondant est m ; sinon $\perp$ est retourné, indiquant que le message chiffré c est invalide.

Le padding PSS utilisé en chiffrement atteint le niveau de sécurité IND-CCA2. De plus, le même jeu de clés peut être utilisé indifféremment pour faire du chiffrement ou de la signature avec le padding PSS [45]. Par ailleurs, ce padding peut également être utilisé pour faire de la «signcryption» [119].

Le fait d'utiliser un même padding et un même jeu de clés pour effectuer différentes opérations cryptographiques présente le double avantage de réduire le code nécessaire à leur programmation et de simplifier les infrastructures (PKIs) nécessaires à leur déploiement.

3.4 Autres constructions

3.4.1 Authentification de séquences

La fonction de hachage SL_2 [149] due à Tillich et Zémor satisfait la propriété intéressante que le haché de la concaténation de deux chaînes binaires est égal au produit des hachés de ces deux chaînes, i.e., $\text{SL}_2(S_1\|S_2) = \text{SL}_2(S_1)\cdot\text{SL}_2(S_2)$. Cette propriété a été exploitée pour authentifier des séquences vidéo [135, 136] en permettant à un éditeur de retirer certaines images d'une séquence (mais sans en altérer l'ordre). Une autre application est l'authentification de séquences vidéo sujettes à des erreurs de transmission. En divisant une image en blocs, l'utilisation de la fonction SL_2 permet de réduire considérablement la probabilité qu'une image ne soit pas authentifiée à cause d'une erreur de transmission [135, 136].

3.4.2 Cryptographie symétrique basée sur l'identité

La cryptographie (à clé publique) basée sur l'identité a été introduite par Shamir en 1984 [143]. L'intérêt de baser la cryptographie sur l'identité est que les certificats ne sont plus nécessaires pour authentifier les clés publiques : dans un cryptosystème à clé publique, l'identité joue le rôle de la clé publique. À partir de l'identité d'un utilisateur, une tierce partie de confiance (TTP) calcule la clé privée correspondante et la fait parvenir de manière sécurisée à l'utilisateur.

Cette idée s'étend naturellement aux systèmes à clé secrète [103]. La clé secrète k_i de l'utilisateur i est dérivée par un TTP à partir de l'identité ID_i de cet utilisateur (et éventuellement d'information auxiliaire h_i) :

$$k_i = H_K\big(ID_i \ \{, h_i\}\big)$$

où H est une fonction à sens unique, paramétrée par la clé-maître K du TTP. Ainsi, de grandes bases de données maintenant la correspondance entre une identité et la clé secrète correspondante ne sont plus nécessaires, ce qui donne lieu à de meilleures performances (gain de mémoire) et à une sécurité accrue (absence d'attaques par dictionnaire). Des exemples concrets d'utilisation sont discutés dans [103].

3.4.3 Génération et diffusion de secrets

Afin d'éviter le rejeu, Lamport a imaginé un système d'authentification par mot de passe «à utilisation unique» [113], basé sur le concept de chaînes à sens unique. À partir d'une valeur initiale S_0 et d'une fonction à sens unique H, un utilisateur calcule de manière itérative $S_i = H(S_{i-1})$ pour $i = 1, \ldots, n$. La valeur finale S_n est donnée de façon sécurisée au serveur. La première fois que l'utilisateur veut se faire authentifier par le serveur, il doit fournir S_{n-1} comme mot de passe et le serveur vérifie que $H(S_{n-1}) = S_n$. Si oui, l'accès est donné à l'utilisateur et le serveur remplace la valeur stockée S_n par S_{n-1}. La fois suivante où l'utilisateur voudra se faire authentifier, il devra fournir la valeur S_{n-2} et le serveur vérifiera que $H(S_{n-2}) = S_{n-1}$ et ainsi de suite jusqu'à ce que la valeur S_0 soit utilisée comme mot de passe.

Ce concept de chaînes à sens unique a été étendu en plusieurs dimensions par les arbres OWCT [102, 106]. Étant donné κ valeurs initiales $(I_1, \ldots, I_\kappa)$ et κ fonctions à sens unique $(H_1, \ldots, H_\kappa)$, les éléments d'un arbre κ-OWCT sont donnés par

$$\big(H_1^{j_1}(I_1), \ldots, H_\kappa^{j_\kappa}(I_\kappa)\big) \quad \text{où } j_1, \ldots, j_\kappa \geq 0 \ .$$

(Le cas $\kappa = 1$ correspond aux chaînes à sens unique.)

Les arbres κ-OWCT avec $\kappa \geq 2$ trouvent de nombreuses applications dans le domaine du dépôt fiduciaire ou de la délégation en minimisant la quantité de données nécessaire pour retrouver des valeurs secrètes et seulement celles-là [106]. Ils permettent enfin de généraliser des schémas précédemment construits avec les chaînes à sens unique (tels des schémas de micro-paiement).

Chapitre 4

Algorithmes efficaces

4.1 Suites de Lucas

Les suites de Lucas trouvent de nombreuses applications dans diverses branches des mathématiques [116, chapitre 6]. En cryptographie, elles sont notamment utilisées pour construire des cryptosystèmes, pour tester la primalité (ou plus exactement la pseudo-primalité) de certains nombres, pour construire des bases normales optimales, pour extraire des racines carrées modulo un premier $p \equiv 1 \pmod 8$, ou encore pour calculer le nombre de points des courbes de KOBLITZ.

Soient P et Q des entiers rationnels et soit $\Delta = P^2 - 4Q$ un non-carré. Si α et β désignent les deux racines de $x^2 - Px + Q = 0$ dans le corps quadratique $\mathbb{Q}(\sqrt{\Delta})$ alors les suites de Lucas $\{V_r\}_{r\geq 0}$ et $\{U_r\}_{r\geq 0}$ avec les paramètres P et Q sont les entiers rationnels satisfaisant

$$V_r(P,Q) = \alpha^r + \beta^r \quad \text{et} \quad U_r(P,Q) = \frac{\alpha^r - \beta^r}{\alpha - \beta} \ .$$

Un algorithme efficace a été re-découvert par YEN et LAIH [158]† pour évaluer rapidemment $V_r(P,1)$. Cet algorithme a ensuite été généralisé à un paramètre Q arbitraire et étendu pour également évaluer $U_r(P,Q)$ ([85] ou annexe A, p. 186ff) :

```
Entrée : P, Q, r = (r_{m-1}, ..., r_0)_2
Sortie : V_r(P,Q)
  R_0 ← 2 ; R_1 ← P ; q_0 ← 1 ; q_1 ← 1
  for j = m − 1 downto 0 do
    q_0 ← q_0 q_1
    if (r_j = 0) then
      q_1 ← q_0 ; R_1 ← R_0 R_1 − P q_0 ; R_0 ← R_0^2 − 2q_0
    else
      q_1 ← Q q_0 ; R_0 ← R_0 R_1 − P q_0 ; R_1 ← R_1^2 − 2q_1
    endif
  endfor
return R_0
```

Fig. 4.1. Évaluation rapide de $V_r(P,Q)$.

† En 1987, MONTGOMERY a proposé un algorithme similaire dans le cadre des courbes elliptiques [122].

En remarquant qu'à la fin de l'algorithme précédent le registre R_1 contient la valeur de $V_{r+1}(P,Q)$ [107], la valeur de $U_r(P,Q)$ peut être obtenue par $U_r = (2V_{r+1} - P\,V_r)/\Delta$. Si la valeur de Δ^{-1} n'est pas disponible ou si son calcul est relativement coûteux, une méthode alternative pour obtenir $U_r(P,Q)$ est donnée dans [85].

Enfin, il est à noter que cet algorithme (Fig. 4.1) a été retenu dans le standard IEEE P1363 pour calculer efficacement $V_r(P,Q)$ [68, § A.2.4].

4.2 Recodage gauche-droite

Pour de nombreux cryptosystèmes, une des opérations les plus coûteuses est l'exponentiation et un grand nombre de méthodes a été proposé pour accélérer cette opération (voir [65] pour un aperçu récent des algorithmes les plus efficaces). Toutefois, seul l'algorithme du «square-and-multiply» (Fig. 4.2-a) et ses quelques variantes sont réellement adaptés aux environnements contraints (tels les cartes à puce par exemple).

Entrée : $\alpha, r = (r_{m-1}, \ldots, r_0)_2$ Sortie : α^r

$R_0 \leftarrow 1$
for i **from** $m-1$ **downto** 0 **do**
  $R_0 \leftarrow {R_0}^2$
  if $(r_i = 1)$ **then** $R_0 \leftarrow R_0 \cdot \alpha$
endfor
return R_0

(a) Square-and-multiply.

Entrée : $\alpha, r = (r'_m, \ldots, r'_0)_{\text{SD2}}$ Sortie : α^r

$R_0 \leftarrow 1$
for i **from** m **downto** 0 **do**
  $R_0 \leftarrow {R_0}^2$
  if $(r'_i \neq 0)$ **then** $R_0 \leftarrow R_0 \cdot \alpha^{r'_i}$
endfor
return R_0

(b) Variante signée.

Fig. 4.2. Algorithmes binaires d'exponentiation.

L'algorithme du square-and-multiply (Fig. 4.2-a) parcourt les bits de l'exposant de la gauche vers la droite. Il existe une variante qui parcourt les bits de l'exposant de la droite vers la gauche mais celle-ci nécessite un registre de travail supplémentaire. Une autre variante (Fig. 4.2-b) prend en entrée une représentation binaire signée de l'exposant :

$$r = \sum_{i=0}^{m} r'_i\, 2^i \quad \text{avec } r'_i \in \{-1, 0, 1\} \ .$$

La représentation (binaire) signée d'un nombre n'est pas unique. Cependant, parmi toutes les représentations signées possibles, on distingue le NAF (de l'anglais «non-adjacent form»), lequel tire son nom du fait que le produit de deux chiffres adjacents d'un NAF est nul : $r'_i \cdot r'_{i+1} = 0$. De façon remarquable, en plus

d'être unique, le NAF a un poids de Hamming (c.-à-d. un nombre de chiffres non nuls) qui est minimal [139]. Cette propriété de minimalité est particulièrement intéressante car elle fait passer la complexité moyenne de l'algorithme du square-and-multiply (Fig. 4.2-a) de $1.5m$ multiplications à $1.33m$ multiplications pour sa variante signée (Fig. 4.2-b) [1], en supposant toutefois que la valeur de α^{-1} est disponible (i.e., précalculée) ou que son calcul est peu coûteux. Ceci est notamment le cas lorsque l'«exponentiation» est effectuée dans le groupe additif des entiers [22] ou dans le groupe des points d'une courbe elliptique [123].

Entrée : $(r_{m-1}, \ldots, r_0)_2$
Sortie : $(r'_m, \ldots, r'_0)_{\text{SD2}}$

$c_0 \leftarrow 0\,;\, r_{m+1} \leftarrow 0\,;\, r_m \leftarrow 0$
for i **from** 0 **to** m **do**
  $c_{i+1} \leftarrow \lfloor (c_i + r_i + r_{i+1})/2 \rfloor$
  $r'_i \leftarrow c_i + r_i - 2c_{i+1}$
endfor

(a) Recodage gauche-droite (NAF).

Entrée : $(r_{m-1}, \ldots, r_0)_2$
Sortie : $(r^\star_m, \ldots, r^\star_0)_{\text{SD2}}$

$b_m \leftarrow 0\,;\, r_m \leftarrow 0\,;\, r_{-1} \leftarrow 0\,;\, r_{-2} \leftarrow 0$
for i **from** m **downto** 0 **do**
  $b_{i-1} \leftarrow \lfloor (b_i + r_{i-1} + r_{i-2})/2 \rfloor$
  $r^\star_i \leftarrow -2b_i + r_i + b_{i-1}$
endfor

(b) Recodage droite-gauche (SF).

Fig. 4.3. Algorithmes de recodage binaire signé minimal.

Il est à noter que l'algorithme de Reitwiesner [139] (Fig. 4.3-a) pour produire le NAF d'un nombre à partir de sa représentation binaire parcourt le nombre à recoder de la droite vers la gauche. Une analyse détaillée de cet algorithme révèle que le calcul du NAF de r revient à retrancher r à $3r$ (avec la règle supplémentaire que $0 - 1 = -1$) et d'ensuite ignorer le dernier (i.e., le moins significatif) 0. Malheureusement, aucun algorithme d'addition droite-gauche (pour le calcul de $3r$) produisant un chiffre du résultat à chaque étape n'est connu, si bien que l'obtention à la volée d'un NAF de la droite vers la gauche semble impossible. Toutefois, en exploitant l'équivalence de certaines formes binaires signées (c.-à-d. des formes représentant le même nombre, ayant la même longueur et le même poids de Hamming), il est possible de lever les ambiguïtés pour obtenir un chiffre à chaque étape. La représentation résultante, appelée SF (de l'anglais «star form»), ne présente pas la propriété de non-adjacence mais, tout comme le NAF, est minimale ([104] ou annexe A, p. 191ff). L'algorithme de recodage droite-gauche correspondant est présenté à la figure 4.3-b. Comparé à l'algorithme de Reitwiesner, cet algorithme présente l'avantage d'être directement compatible avec la variante signée de l'algorithme du square-and-multiply (Fig. 4.2-b) : aucun pré-stockage n'est donc nécessaire.

Des représentations minimales (gauche-droite) pour une base arbitraire $b \geq 2$, généralisant les NAFs, ont été étudiées par Clark et Liang [35]. L'obtention droite-gauche de représentations minimales pour une base arbitraire n'est cependant connue que depuis peu [105] (ou annexe A, p. 209ff).

En plus de leur utilité aux techniques d'arithmétique rapide, ces représentations minimales trouvent également des applications en théorie des codes [153].

4.3 Génération de nombres premiers

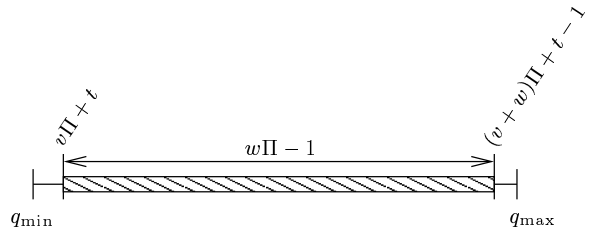
Une méthode simple pour générer un nombre premier aléatoire q dans un intervalle $[q_{\min}, q_{\max}]$ consiste à tirer un nombre aléatoire dans $[q_{\min}, q_{\max}]$ et d'en tester la primalité. Afin de minimiser le nombre de candidats à tester, une meilleure stratégie consiste à ne considérer que les candidats pairs ou plus généralement que les candidats relativement premiers avec $\Pi := \prod p_i$, avec p_i premier. En cas de test infructueux, le candidat q est alors remplacé par $q \leftarrow q + \Pi$, de sorte que le nouveau candidat reste relativement premier avec Π [23]. Une méthode optimisant ce principe est donnée dans [82] (ou annexe A, p. 218ff). Nous en présentons ci-dessous une légère généralisation, telle que donnée dans [80].

Soit un paramètre de qualité $0 < \epsilon \leq 1$ (typiquement $\epsilon = 10^{-3}$). La phase d'initialisation de l'algorithme nécessite un produit de premiers $\Pi = \prod_i p_i$ tel qu'il existe des entiers t, v et w satisfaisant :

(P1) $1 - \epsilon < \dfrac{w\Pi - 1}{q_{\max} - q_{\min}} \leq 1$;

(P2) $v\Pi + t \geq q_{\min}$ et $(v + w)\Pi + t - 1 \leq q_{\max}$; et

(P3) le rapport $\phi(\Pi)/\Pi$ est aussi petit que possible.



Il est à noter que l'algorithme génère des nombres premiers dans le sous-intervalle $[v\Pi + t, (v + w)\Pi + t - 1] \subseteq [q_{\min}, q_{\max}]$ (propriété (P2)). Le paramètre ϵ reflète l'erreur faite par cette approximation (propriété (P1)). La minimalité du rapport $\phi(\Pi)/\Pi$ assure que Π contient un maximum de premiers (propriété (P3)).

Entrée : paramètres $l = v\Pi$, $m = w\Pi$, t, et $a \in (\mathbb{Z}/m\mathbb{Z})^*$
Sortie : premier $q \in [q_{\min}, q_{\max}]$

$k \in_R (\mathbb{Z}/m\mathbb{Z})^*$
$q \leftarrow [(k - t) \bmod m] + t + l$
while (q n'est pas premier) **do**
  $k \leftarrow ak \pmod m$
  $q \leftarrow [(k - t) \bmod m] + t + l$
endwhile
return q

Fig. 4.4. Algorithme de génération de premiers.

Comme $a, k \in (\mathbb{Z}/m\mathbb{Z})^*$, il s'ensuit que $ak \in (\mathbb{Z}/m\mathbb{Z})^*$ et donc que $\text{pgcd}(q, \Pi) = \text{pgcd}(k, \Pi) = 1$, après la mise à jour $k \leftarrow ak \pmod m$. La probabilité que le candidat q est premier est par conséquent élevée.

Notons par ailleurs qu'il existe une variante hors-ligne/en-ligne de cet algorithme [55, 56]. Dans cette variante, un grand nombre de premiers peut être généré à la volée (i.e., en ligne) en un temps réduit.

Des techniques efficaces permettant de générer des premiers vérifiant des propriétés additionnelles sont présentées dans [82] (ou annexe A, p. 218ff). À titre d'exemple, nous montrons brièvement comment générer un premier DSA, c.-à-d. un premier $p \in [p_{\min}, p_{\max}]$ de la forme $p = 1 + qr$ où q est un premier de 160 bits. La première étape consiste à générer un premier q de 160 bits. Ensuite, nous définissons $\Pi = \prod_{p_i \neq q} p_i$ et choisissons r de sorte que

$$r \equiv -\frac{1}{q} + c \pmod \Pi$$

avec c relativement premier avec Π. Il est alors aisé de vérifier que $\text{pgcd}(p, \Pi) = \text{pgcd}(qc, \Pi) = 1$.

L'algorithme de génération de premiers (Fig. 4.4) nécessite de tirer aléatoirement une unité k dans $(\mathbb{Z}/m\mathbb{Z})^*$. Une méthode constructive pour générer une unité est donnée par la proposition suivante.

Proposition 1 ([80]). $\forall k, r \in \mathbb{Z}/m\mathbb{Z}$ *tels que* $\text{pgcd}(r, k, m) = 1$*, nous avons*

$$[k + r(1 - k^{\lambda(m)})] \in (\mathbb{Z}/m\mathbb{Z})^* \ .$$

□

Nous obtenons ainsi :

Entrée : m
Sortie : $k \in (\mathbb{Z}/m\mathbb{Z})^*$

```
k ∈R [1, m[
U ← 1 − k^λ(m) (mod m)
while (U ≠ 0) do
   r ∈R [1, m[
   k ← k + rU (mod m) ; U ← 1 − k^λ(m) (mod m)
endwhile
return k
```

Fig. 4.5. Algorithme de génération d'unités.

La condition sur U permet de tester que k est inversible modulo m ; en effet, $k \in (\mathbb{Z}/m\mathbb{Z})^* \iff k^{\lambda(m)} \equiv 1 \pmod m$ ([80, Proposition 1]).

L'algorithme présenté à la figure 4.5 a la propriété d'être «auto-correcteur». Par la proposition 1, dès que k est relativement premier avec un facteur de m, il reste premier avec ce facteur après l'étape de mise à jour, $k \leftarrow k + rU \pmod m$.

4.4 Autres algorithmes

4.4.1 Normalisation DR

La multiplication modulaire est l'opération de base de la plupart des cryptosystèmes à clé publique. Par exemple, avec le RSA, la signature d'un message m est donnée par $s = \mu(m)^d \bmod n$ où μ est un padding approprié et d est l'exposant de signature. Dans l'hypothèse où la multiplication modulo $n' = \delta \cdot n$ est plus rapide, la signature s peut être obtenue plus efficacement comme

$$s = \frac{\big(\delta(\mu(m)^d \bmod n') \bmod n'\big)}{\delta} \ .$$

Cette idée a été exploitée indépendamment par Quisquater [134] et Walter [154]. Dans ces méthodes, le module «normalisé» n' a ses plus bits les plus significatifs égaux à 1.

Soient $n = \sum_{i=0}^{\nu-1} n_i\, 2^i$ et $n' = \sum_{i=0}^{\nu'-1} n'_i\, 2^i$. Le module normalisé n' est dit de type DR (de l'anglais «diminished radix») d'ordre c si $n' = \delta\, n = 2^{\nu'} - \mu$ avec $\mu < 2^\nu$ et $\nu' = \nu + c$ [131]. Dans l'algorithme de Quisquater, le facteur de normalisation est donné par $\delta_{_Q} := \lfloor 2^{\nu'}/n \rfloor$. Il n'est cependant pas nécessaire d'effectuer la division complète par n pour obtenir $\delta_{_Q}$ [72, 51]. En effet, si

$$\hat{n} := \sum_{i=\nu-c-2}^{\nu-1} n_i\, 2^{i-k} \quad \text{et} \quad \hat{\delta}_{_Q} := \left\lfloor \frac{2^{2c+2}}{\hat{n}} \right\rfloor ,$$

alors $\delta_{_Q} \leq \hat{\delta}_{_Q} \leq \delta_{_Q} + 1$. Ainsi, avec au plus une correction, la valeur exacte de $\delta_{_Q}$ peut être obtenue à partir des $(c+2)$ bits de poids fort de n.

D'autres techniques pour améliorer les performances du RSA sont présentées dans [81].

4.4.2 Encodage compact de NAFs

L'algorithme de Reitwiesner [139] (Fig. 4.3-a) exprime un nombre r sous forme de NAF, $r = \sum_{i=0}^{m} r'_i\, 2^i$ avec $r'_i \in \{-1, 0, 1\}$. Étant donné le signe '$-$', chaque chiffre r'_i est habituellement encodé sur deux bits : le bit de signe suivi du bit de valeur. Ainsi, la représentation de Reitwiesner nécessite deux fois plus de mémoire pour son stockage que la représentation binaire. Il est cependant possible d'exploiter la propriété de non-adjacence du NAF (c.-à-d. $r'_i \cdot r'_{i+1} = 0$, $\forall i$) pour encoder le nombre r avec la même efficacité (à un bit près) que la représentation binaire [99].

Par la propriété de non-adjacence, chaque chiffre '1' ou '-1' de la représentation en NAF est suivi d'un '0'. Ceci suggère le simple encodage droite-gauche suivant :

$$\mathcal{R} : \begin{cases} 01 \mapsto 01 \\ 0\bar{1} \mapsto 11 \\ 0 \mapsto 0 \end{cases} \qquad \text{et} \qquad \mathcal{R}^{-1} : \begin{cases} 01 \mapsto 01 \\ 11 \mapsto 0\bar{1} \\ 0 \mapsto 0 \end{cases} \ .$$

Avec cette conversion $\mathcal{R}$, la représentation d'un NAF ne nécessite qu'un bit de plus que la représentation binaire. Par exemple, une application droite-gauche de l'encodage $\mathcal{R}$ à $\text{NAF}(29) = (100\bar{1}01)$ donne 101101 :

$$(\underline{1}\,\underline{0}\,\underline{0\bar{1}}\,\underline{01}) \mapsto (\underline{1}\,\underline{0}\,\underline{11}\,\underline{01})$$

et la transformation inversion $\mathcal{R}^{-1}$ redonne la forme NAF.[‡]

Il est également possible d'encoder efficacement les NAFs de la gauche vers la droite ou encore d'aplliquer ces techniques aux τ-NAFs [65] utilisés pour accélérer la multiplication scalaire sur les courbes de Koblitz [99].

Enfin, une légère modification de la transformation $\mathcal{R}^{-1}$ permet de tirer des nombres aléatoires de m bits sous forme non-adjacente.

La première étape consiste à tirer un nombre aléatoire dans $\{0,1\}^m$ et de lui appliquer la transformation droite-gauche $\mathcal{R}^{-1}$. Si le chiffre le plus significatif de la représentation ainsi obtenue est '-1' alors cette représentation est complétée par des '0' jusque la position $m-1$ et par un '1' à la position m. L'ensemble de ces deux opérations définit la représentation $\mathcal{R}^*$.

Il est aisé de montrer que :

Proposition 2 ([99]). *La transformation $\mathcal{R}^*$ induit une permutation sur l'ensemble $\{\text{NAF}(k) \mid k \in \{0,1\}^m\}$.* □

À titre d'illustration, la table suivante donne respectivement le NAF et la représentation $\mathcal{R}^*$ des nombres de 3 bits.

k	$\text{NAF}(k)$	$\mathcal{R}^*(k)$
0	(0)	(0)
1	(1)	(1)
2	(10)	(10)
3	$(10\bar{1})$	$(100\bar{1})$
4	(100)	(100)
5	(101)	(101)
6	$(10\bar{1}0)$	$(10\bar{1}0)$
7	$(100\bar{1})$	$(10\bar{1})$

4.4.3 Multiplication scalaire sur courbes elliptiques

L'opération de base des cryptosystèmes basés sur les courbes elliptiques est la multiplication scalaire : étant donné un point $\boldsymbol{P}$ et un entier k, il faut calculer le point $\boldsymbol{Q} := k\boldsymbol{P} = \boldsymbol{P}+\boldsymbol{P}+\cdots+\boldsymbol{P}$ (k fois). En grande caractéristique, une courbe elliptique E sur un corps premier $\mathbb{F}_p$ est donnée par l'équation de Weierstraß

$$E_{/\mathbb{F}_p} : y^2 = x^3 + ax + b \ .$$

Pour des raisons d'efficacité, les points de la courbe sont représentées en coordonnées jacobiennes et le paramètre a est choisi égal à -3 [49]. Lorsque

[‡] Pour plus de lisibilité, $\bar{1}$ est écrit pour -1.

$a \neq -3$, on peut s'y ramener par isomorphisme avec une probabilité de $1/2$ quand $p \equiv 3 \pmod 4$ et avec une probabilité de $1/4$ quand $p \equiv 1 \pmod 4$ [68]. Dans les autres cas, on peut travailler sur une courbe isogène (de petit degré) ayant un paramètre $a = -3$ [25].

Si φ dénote une isogénie de petit degré m (avec $\mathrm{pgcd}(m, \mathrm{ord}_E(\boldsymbol{P})) = 1$)[§] entre les courbes $E_{/\mathbb{F}_p} : y^2 = x^3 + ax + b$ et $E'_{/\mathbb{F}_p} : y^2 = x^3 - 3x + b'$, alors l'évaluation $\boldsymbol{Q} = k\boldsymbol{P}$ peut se faire comme $\boldsymbol{Q} = \hat{\varphi}\big(k_m\, \varphi(\boldsymbol{P})\big)$ où $k_m = k/m \bmod \mathrm{ord}_E(\boldsymbol{P})$ et $\hat{\varphi}$ est l'isogénie duale de φ :

$$\begin{array}{ccc} \boldsymbol{P} \in E(\mathbb{F}_p) & \xrightarrow{\quad [k] \quad} & \boldsymbol{Q} = k\boldsymbol{P} \in E(\mathbb{F}_p) \\ \varphi \big\downarrow & & \big\uparrow \hat{\varphi} \\ \boldsymbol{P}' \in E'(\mathbb{F}_p) & \xrightarrow{\quad [k_m]' \quad} & \boldsymbol{Q}' = k_m\boldsymbol{P}' \in E'(\mathbb{F}_p) \end{array}$$

Comme la courbe E' a son paramètre $a' = -3$, le calcul de $\boldsymbol{Q}' = k_m\boldsymbol{P}'$ est plus rapide. Un exemple concret d'application avec des tailles cryptographiques est donné dans [25, annexe A].

Par ailleurs, cette technique illustre le fait que choisir des courbes elliptiques avec $a = -3$ (telles celles recommandées dans la plupart des standards cryptographiques) n'est pas un choix restrictif.

[§] Cette condition est automatiquement satisfaite pour les courbes utilisées en cryptographie car le point de base $\boldsymbol{P}$ de ces courbes est d'ordre un grand premier (typiquement de 160 bits ou plus).

Chapitre 5

Implantations sécurisées

5.1 Attaques par fautes

La sécurité prouvée (voir section 2.4) permet de valider la résistance d'un cryptosystème dans un modèle de sécurité donné. Les modèles de sécurité actuels considèrent des adversaires ayant un accès en «boîte noire» ; en particulier, l'implantation d'un cryptosystème est supposée «sécurisée» et son exécution est supposée correcte. La première hypothèse sous-entend une résistance contre les attaques à canaux cachés (voir section 5.2) alors que la deuxième hypothèse sous-entend une résistance contre les attaques par fautes.

En septembre 1996, des chercheurs de Bellcore annonçaient une nouvelle attaque [8] (plus tard publiée dans [19, 20]) contre le RSA et d'autres systèmes à clé publique. Cette attaque exploite la présence de fautes (intentionnelles ou non) lors de calculs cryptographiques. Nous donnons ci-dessous une version simplifiée de l'attaque de Bellcore, telle que présentée dans [79] (ou annexe A, p. 233ff).

Soit $n = pq$ un module de type RSA. La signature d'un message m avec le RSA est donnée par $s = \mu(m)^d \bmod n$ où μ est une fonction de padding et d est l'exposant secret de signature. Le calcul de la signature s peut être accéléré par CRT (de l'anglais «Chinese Remainder Theorem») : si $x = \mu(m)$ alors $s = \text{CRT}(s_p, s_q)$ avec $s_t := {x_t}^{d_t} \bmod t$, $x_t := x \bmod t$ et $d_t := d \bmod (t-1)$ $(t \in \{p, q\})$.

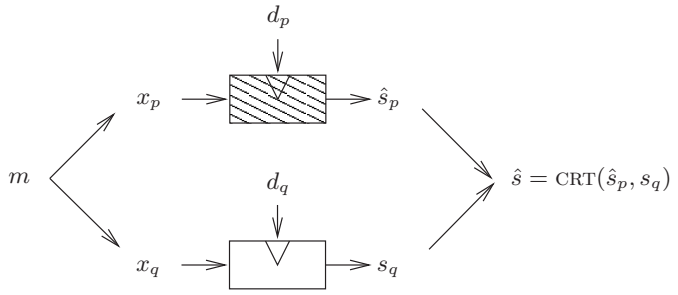


Fig. 5.1. Calcul (erroné) d'une signature RSA par CRT.

Si une erreur se produit lors du calcul de s_p (nous notons $\hat{s}_p$ la valeur erronée) mais que le calcul de s_q est correct, alors

$$\text{pgcd}\big(\hat{s}^e - \mu(m) \pmod n, n\big) = q$$

et le module RSA est factorisé [79]. Cette attaque suppose que la fonction de padding μ est déterministe ; en particulier, elle ne s'applique pas au schéma de signature RSA-PSS [7, 141]. Cependant, moyennant des hypothèses d'attaques plus fortes, les schémas de signature probabilistes peuvent également être sujet à une attaque similaire. Par exemple, si un attaquant est capable de forcer l'exposant CRT d_p à zéro alors $\text{pgcd}(\hat{s} - 1, n) = p$.

Un autre modèle d'attaque, proposé par Bao *et al.* [4], suppose qu'un attaquant bascule la valeur d'un bit de l'exposant de signature RSA (modèle transitoire). Une version simplifiée de cette attaque est décrite dans [93]. Dans les deux cas, en supposant que la fonction de padding est déterministe, il est facile de retrouver la valeur de ce bit.

Cette attaque ainsi que la précédente s'applique à de nombreux systèmes de type RSA [93, 79]. D'autres attaques par fautes contre le RSA ainsi que leur mise en œuvre sont discutées dans [64, 157, 9]. Les attaques par fautes ont également été étendues aux systèmes à clé secrète par Biham et Shamir [11].

Une parade efficace pour contrer les attaques par fautes consiste à effectuer les calculs de façon redondante. Une contre-mesure concrète a été proposée par Shamir dans le cadre du RSA [145]. Cette contre-mesure a ensuite été étendue à tout système à base d'arithmétique modulaire dans [83].

5.2 Attaques par canaux cachés

Une deuxième classe d'attaques contre les implantations concerne les attaques par canaux cachés. Ces attaques ont été introduites par Kocher [110, 111]. Deux types d'attaques par canaux cachés peuvent être distinguées : les attaques de type simple (par exemple la SPA) et les attaques de type différentiel (par exemple la DPA).

Une attaque de type simple consiste à observer un canal caché (comme le temps d'exécution, la consommation en courant ou le rayonnement électromagnétique) et d'en déduire de l'information sur les données secrètes. La figure suivante représente la consommation en courant d'une exponentiation modulaire effectuée par une implantation naïve de l'algorithme du square-and-multiply (Fig. 4.2-a).

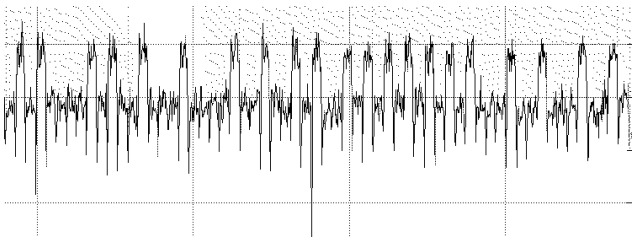


Fig. 5.2. Trace en courant de l'algorithme du square-and-multiply.

Si l'on sait que les niveaux de basse consommation correspondent à des carrés et que les niveaux de haute consommation correspondent à des multiplications, il est facile d'en déduire la valeur de l'exposant.

Une façon simple pour se protéger contre cette attaque est d'effectuer une multiplication factice dans le cas où le bit de l'exposant est un zéro [41]. L'algorithme «régulier» ainsi obtenu est appelé «square-and-multiply 'always'» (voir Fig. 5.3-a). Cet algorithme nécessite $2m$ multiplications au lieu des $\frac{3}{2}m$ multiplications, en moyenne, de l'implantation naïve. Une autre possibilité est d'utiliser l'algorithme d'exponentiation de MONTGOMERY [122]. Ce dernier est régulier sans introduire d'opérations factices et est par conséquent résistant à certaines attaques par fautes [107]. Sa complexité est cependant également de $2m$ multiplications. Un exemple d'algorithme ayant une même complexité moyenne de $\frac{3}{2}m$ multiplications est donné par la figure 5.3-b ([74]) ; d'autres algorithmes d'exponentiation réguliers et efficaces sont présentés à la section 5.3. Un algorithme de division protégé contre les attaques de type SPA est donné dans [101].

Entrée : $\alpha, r = (r_{m-1}, \ldots, r_0)_2$ Sortie : α^r

$R_0 \leftarrow 1$
for i **from** $m-1$ **downto** 0 **do**
  $R_0 \leftarrow {R_0}^2$
  $b \leftarrow \neg r_i$; $R_b \leftarrow R_b \cdot \alpha$
endfor
return R_0

(a) Square-and-multiply 'always'.

Entrée : $\alpha, r = (r_{m-1}, \ldots, r_0)_2$ Sortie : α^r

$R_0 \leftarrow 1$; $R_1 \leftarrow \alpha$; $i \leftarrow 0$
while $(i \leq m-1)$ **do**
  $b \leftarrow \neg r_i$
  $R_b \leftarrow R_b \cdot R_1$; $r_i \leftarrow 0$; $i \leftarrow i + b$
endwhile
return R_0

(b) «Ré-écriture» de l'exposant.

Fig. 5.3. Algorithmes binaires d'exponentiation protégés contre la SPA.

Une approche plus formelle de la résistance contre les attaques simples de type SPA est entreprise dans [36] (ou annexe A, p. 238ff). L'idée consiste à «transférer» la sécurité de la routine d'exponentiation dans l'exposant lui-même (lequel est une donnée secrète). Cela est fait à l'aide de chaînes d'addition ou plus exactement de «chaînes de registres» [36].

Soit $\mathcal{C}(r) = \{r^{(0)}, r^{(1)}, \ldots, r^{(m)}\}$ une chaîne d'addition pour l'exposant r :

(C1) $r^{(0)} = 1$; et

(C2) $\forall 1 \leq i \leq m, \exists\, j(i), k(i) < i$ tels que $r^{(i)} = r^{(j(i))} + r^{(k(i))}$.

Alors la chaîne de registres correspondante est définie par

$$\Gamma(r) = \Big\{ \big(w(i); u(i), v(i)\big) \Big\}_{1 \leq i \leq m} ,$$

signifiant qu'à l'étape i, les valeurs de $\alpha^{j(i)}$ et $\alpha^{k(i)}$ se trouvent respectivement dans les registres $R_{u(i)}$ et $R_{v(i)}$ et que leur produit $\alpha^{r(i)} = \alpha^{j(i)} \cdot \alpha^{k(i)}$ doit être

Entrée : $\alpha, \Gamma(r)$
Sortie : α^r

$R_{u(1)} \leftarrow \alpha$; $R_{v(1)} \leftarrow \alpha$
for i **from** 1 **to** m **do** $R_{w(i)} \leftarrow R_{u(i)} \cdot R_{v(i)}$
return $R_{w(m)}$

Fig. 5.4. Algorithme universel d'exponentiation.

écrit dans le registre $R_{w(i)}$. Étant donné une chaîne de registres $\Gamma(r)$, le calcul de α^r devient trivial (voir Fig. 5.4). La sécurité de l'implantation contre les attaques de type SPA est validée si l'opération «atomique» $R_{w(i)} \leftarrow R_{u(i)} \cdot R_{v(i)}$ ne laisse fuir aucune information «utile» par mesure de courant. Cette méthodologie (i.e., réduction) est semblable aux preuves de sécurité de la cryptographie moderne, si ce n'est que la sécurité n'est pas réellement prouvée (mais plutôt observée) et que le même travail d'observation doit être fait pour chaque canal caché donné.

Le désavantage majeur de l'algorithme universel (Fig. 5.4) est que la représentation $\Gamma(r)$ est non-standard. Cela signifie que si $\Gamma(r)$ n'est pas donné en entrée de l'algorithme, il doit être lui-même calculé de façon à résister contre les attaques de type SPA. Des solutions à ce problème sont discutées dans [36]. Il est à noter qu'une de ces solutions a donné lieu à l'algorithme MIST [155].

Les attaques différentielles de type DPA peuvent être évitées en rendant aléatoire l'exécution de l'algorithme cryptographique. Typiquement, une signature RSA, $s = \mu(m)^d \bmod n$, se calcule de la façon suivante :

$$s = \left[(\mu(m) + r_1\, n)^{d + r_2\, \phi(n)} \bmod (r_3\, n)\right] \bmod n$$

où r_1, r_2, r_3 sont des nombres aléatoires de 32 ou 64 bits. Un traitement plus formel de la résistance aux attaques différentielles par canaux cachés peut être trouvé dans [28].

5.3 Atomicité

Il est relativement aisé, du moins en cryptographie à clé publique, de se prémunir contre les attaques de type DPA en rendant aléatoire l'exécution de l'algorithme cryptographique. La prévention des attaques de type SPA est plus délicate car elle est fortement liée à l'architecture sur laquelle est implanté l'algorithme cryptographique. Une méthode simple pour prévenir les attaques de type SPA pour l'exponentiation (modulaire) consiste à ajouter une multiplication factice dans l'algorithme du square-and-multiply lorque le bit de l'exposant est égal à zéro (voir Fig.5.3-a). L'«atomicité par canal caché» [31] (ou annexe A, p. 248ff) raffine autant que possible cette approche en ajoutant un minimum d'opérations factices de sorte que le code ne laisse fuir aucune information sensible par SPA (ou toute autre attaque de type simple), ce qui résulte en une vitesse d'exécution accrue.

Un bloc atomique commun par canal caché Γ pour un ensemble de processus $\{\Pi_0, \ldots, \Pi_n\}$ est une séquence d'instructions telle que chacun des processus Π_j puisse être exprimé comme une répétition de séquences d'instructions équivalentes à Γ ; deux séquences d'instructions étant équivalentes si elles sont indistinguables par canal caché. En d'autres mots, il existe des séquences d'instructions $\gamma_{j,i} \sim \Gamma$ telles que $\Pi_j = \gamma_{j,1} \| \gamma_{j,2} \| \cdots \| \gamma_{j,\ell_j}$. Il est à noter qu'un bloc atomique commun Γ existe toujours par l'ajout éventuel d'instructions factices dans une séquence d'instructions $\gamma_{j,i} \sim \Gamma$. Par exemple, dans l'algorithme du «square-and-multiply 'always'» (Fig.5.3-b), le bloc atomique commun Γ correspond au choix trivial d'un carré, d'une multiplication et d'une décrémention de compteur. Ce choix est sous-optimal ; un bloc commun plus «atomique» est donné par la figure suivante.

while $(i \geq 0)$ **do**
  case
    $(r_i = 1)$: /* Processus Π_0 */
      $R_0 \leftarrow R_0 \cdot R_0$: ι_0
      $i \leftarrow i - 0$: $\iota_2^{(\text{factice})}$ } $= \gamma_0 \sim \Gamma$
      $R_0 \leftarrow R_0 \cdot R_1$: ι_1
      $i \leftarrow i - 1$: ι_2 } $= \gamma_1 \sim \Gamma$
    $(r_i = 0)$: /* Processus Π_1 */
      $R_0 \leftarrow R_0 \cdot R_0$: ι_3
      $i \leftarrow i - 1$: ι_4 } $= \gamma_2 \sim \Gamma$
  endcase
endwhile

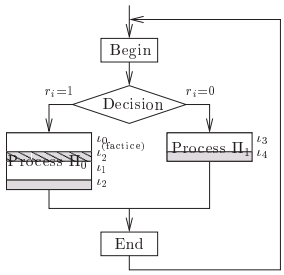


Fig. 5.5. Bloc atomique commun Γ pour l'algorithme du square-and-multiply.

Une fois le bloc atomique commun Γ identifié, la deuxième étape consiste à enchaîner les différentes séquences $\gamma_{j,i}$ pour que l'ensemble du code apparaisse comme une succession régulière de blocs $\gamma_k = \gamma_{j,i} \sim \Gamma$, indistinguables par analyse de type SPA. Pour cela, un bit s est utilisé pour savoir s'il y a encore des séquences à traiter au sein du processus Π_j et le compteur de séquences k est mis à jour par

$$k \leftarrow (\neg s) \cdot (k + 1) + s \cdot f(\text{valeurs d'entrée}) \ .$$

Avec l'exemple précédent, nous obtenons la table :

	k	s	γ_k
$(r_i = 1)$	0	0	$R_0 \leftarrow R_0 \cdot R_0 \, ; \, i \leftarrow i - 0$
	1	1	$R_0 \leftarrow R_0 \cdot R_1 \, ; \, i \leftarrow i - 1$
$(r_i = 0)$	2	1	$R_0 \leftarrow R_0 \cdot R_0 \, ; \, i \leftarrow i - 1$

Lorsque les blocs γ_k sont relativement simples, ils peuvent être explicités directement comme des fonctions de k et de s. Ainsi appliqué à l'algorithme du

square-and-multiply et à sa variante droite-gauche, nous obtenons après simplification les algorithmes suivants :

Entrée : $\alpha, r = (r_{m-1}, \ldots, r_0)_2$ Sortie : α^r

$R_0 \leftarrow 1$; $R_1 \leftarrow \alpha$; $i \leftarrow m-1$; $b \leftarrow 0$
while $(i \geq 0)$ **do**
$R_0 \leftarrow R_0 \cdot R_b$
$b \leftarrow b \oplus r_i$; $i \leftarrow i - 1 + b$
endfor
return R_0

(a) Gauche-droite.

Entrée : $\alpha, r = (r_{m-1}, \ldots, r_0)_2$ Sortie : α^r

$R_0 \leftarrow 1$; $R_1 \leftarrow \alpha$; $i \leftarrow 0$; $b \leftarrow 1$
while $(i \leq m-1)$ **do**
$b \leftarrow b \oplus r_i$
$R_b \leftarrow R_b \cdot R_1$; $i \leftarrow i + b$
endwhile
return R_0

(b) Droite-gauche.

Fig. 5.6. Algorithmes binaires d'exponentiation protégés contre la SPA (II).

Lorsque les blocs γ_k sont plus compliqués, le même principe peut être appliqué en exprimant de façon implicite ces blocs par une matrice.

Des exemples d'application supplémentaires, notamment à l'algorithme d'exponentiation (M, M^3) [120, Algorithme 14.85] ou à la multiplication de points sur courbe elliptique, sont décrits dans [31].

5.4 Courbes elliptiques

Au vu de la règle de la «sécante-tangente», l'addition et le doublement de points sur courbe elliptique sont des opérations intrinsèquement distinctes. Ainsi, l'algorithme universel d'exponentiation (Fig. 5.4) ou les versions additives des algorithmes d'exponentiation sécurisée donnés à la figure 5.6 ne peuvent pas être utilisés pour évaluer une multiplication scalaire sur courbe elliptique, résistante contre les attaques de type SPA [76].

Il y a trois approches possibles pour éviter les attaques de type SPA :

1. insérer des opérations factices [41, 31] ;
2. considérer des paramétrisations alternatives à la forme de Weierstrass [91, 115, 12] ou «unifier» les formules d'addition [24, 31] ;
3. utiliser des algorithmes qui se comportent par défaut de façon régulière [117, 130, 24, 69, 74, 31].

Ces approches sont bien évidemment complémentaires. Les première et troisième approches partagent le même but : obtenir un algorithme «régulier» quelles que soient les données en entrée. Les algorithmes réguliers communément utilisés pour la multiplication scalaire sur courbe elliptique sont l'algorithme du «double-and-add 'always'» [41] et l'algorithme d'exponentation de Montgomery [122, 107]. Ce dernier algorithme a été détaillé pour les courbes elliptiques

en caractéristique 2 par López et Dahab [117] et en grande caractéristique première par Okeya et Sakurai [130] (pour les courbes de Montgomery) ; sa généralisation à des courbes elliptiques arbitraires est donnée dans [24, 69]. D'autres algorithmes, plus efficaces, basés sur le principe d'atomicité par canal caché (voir section 5.3) sont décrits dans [31].

La deuxième approche nécessite d'avoir des formules indistinguables pour l'addition et le doublement pour ensuite les utiliser dans des algorithmes «atomiques» ([36, 74, 31]). En caractéristique 2, cela peut s'obtenir très simplement en ajoutant deux additions (dans le corps de définition) factices [31] pour le doublement :

Entrée : $(T_1, T_2) = \boldsymbol{P_1}$, $(T_3, T_4) = \boldsymbol{P_2}$
Sortie : $\boldsymbol{P_1} + \boldsymbol{P_2}$ ou $2\boldsymbol{P_1}$

Addition : $\boldsymbol{P_1} \leftarrow \boldsymbol{P_2} + \boldsymbol{P_1}$		*Doublement* : $\boldsymbol{P_1} \leftarrow 2\boldsymbol{P_1}$	
$T_1 \leftarrow T_1 + T_3$	$(= x_1 + x_2)$	$T_6 \leftarrow T_1 + T_3$	(factice)
$T_2 \leftarrow T_2 + T_4$	$(= y_1 + y_2)$	$T_6 \leftarrow T_3 + T_6$	$(= x_1)$
$T_5 \leftarrow T_2/T_1$	$(= \lambda)$	$T_5 \leftarrow T_2/T_1$	$(= y_1/x_1)$
$T_1 \leftarrow T_1 + T_5$		$T_5 \leftarrow T_1 + T_5$	$(= \lambda)$
$T_6 \leftarrow {T_5}^2$	$(= \lambda^2)$	$T_1 \leftarrow {T_5}^2$	$(= \lambda^2)$
$T_6 \leftarrow T_6 + a$	$(= \lambda^2 + a)$	$T_1 \leftarrow T_1 + a$	$(= \lambda^2 + a)$
$T_1 \leftarrow T_1 + T_6$	$(= x_3)$	$T_1 \leftarrow T_1 + T_5$	$(= x_3)$
$T_2 \leftarrow T_1 + T_4$	$(= x_3 + y_2)$	$T_2 \leftarrow T_1 + T_2$	$(= x_3 + y_1)$
$T_6 \leftarrow T_1 + T_3$	$(= x_2 + x_3)$	$T_6 \leftarrow T_1 + T_6$	$(= x_1 + x_3)$
$T_5 \leftarrow T_5 \cdot T_6$		$T_5 \leftarrow T_5 \cdot T_6$	
$T_2 \leftarrow T_2 + T_5$	$(= y_3)$	$T_2 \leftarrow T_2 + T_5$	$(= y_3)$

return (T_1, T_2)

Fig. 5.7. Formules d'addition indistinguables sur la courbe elliptique (non-supersingulière) d'équation $E_{/\mathbb{F}_{2^q}} : y^2 + xy = x^3 + ax^2 + b$.

Une autre possibilité consiste à unifier directement les formules d'addition.

Proposition 1 ([24]). *Soit une courbe elliptique E définie sur un corps $\mathbb{K}$, donnée par l'équation $E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$. Soient également les points $\boldsymbol{P} = (x_1, y_1)$ et $\boldsymbol{Q} = (x_2, y_2) \in E(\mathbb{K}) \setminus \{\boldsymbol{O}\}$ avec $\mathrm{y}(\boldsymbol{P}) \neq \mathrm{y}(-\boldsymbol{Q})$. Alors $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ où $x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$, $y_3 = -(\lambda + a_1)x_3 - \mu - a_3$ avec*

$$\lambda = \frac{x_1^2 + x_1x_2 + x_2^2 + a_2x_1 + a_2x_2 + a_4 - a_1y_1}{y_1 + y_2 + a_1x_2 + a_3}$$

et $\mu = y_1 - \lambda x_1$. □

Dans [115], Liardet et Smart proposent de représenter les courbes elliptiques comme l'intersection de deux surfaces quadriques. Contrairement à la

paramétrisation de WEIERSTRASS, les mêmes formules d'addition sont valables pour l'addition et le doublement [33]. Cependant, seules les courbes ayant trois points rationnels d'ordre 2 donnent lieu à une arithmétique rapide [115]. Une méthode plus efficace en vitesse et en mémoire pour ce type de courbes elliptiques, utilisant un modèle quartique, est donnée dans [12]. Pour les courbes elliptiques ayant un point rationnel d'ordre 3, le modèle de HESSE peut être utilisé en notant que le doublement d'un point peut s'exprimer comme l'addition de deux points distincts [91] (ou annexe A, p. 263ff).

Proposition 2 ([91]). *Soit un point* $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ *sur la courbe elliptique de* HESSE *donnée par* $E_{/\mathbb{K}} : X^3 + Y^3 + Z^3 = 3DXYZ$. *Alors*

$$2(X_1 : Y_1 : Z_1) = (Z_1 : X_1 : Y_1) + (Y_1 : Z_1 : X_1) \ .$$

De plus, nous avons $(Z_1 : X_1 : Y_1) \neq (Y_1 : Z_1 : X_1)$. □

Tout comme pour les autres algorithmes, les algorithmes cryptographiques sur courbe elliptique se protègent contre les attaques de type DPA en rendant aléatoire les entrées de l'algorithme. En plus des méthodes usuelles, la richesse mathématique des courbes elliptiques peut être exploitée pour fournir davantage de contre-mesures. Ainsi, dans le calcul de $k\boldsymbol{P}$, le point $\boldsymbol{P}$ peut être masqué par l'utilisation de coordonées projectives aléatoires [41] ou encore d'isomorphismes aléatoires de courbes elliptiques ou du corps de définition [100] (ou annexe A, p. 272ff). Dans le cas particulier des courbes de KOBLITZ, l'endomorsphime de FROBENIUS peut être utilisé pour masquer la valeur du multiplieur k [100].

Une étude approfondie de la résistance des cryptosystèmes sur courbe elliptique aux attaques par fautes, complétant une analyse précédente de BIEHL, MEYER et MÜLLER [10] est faite dans [34]. La conclusion est qu'en plus des données secrètes, les paramètres publics doivent également être protégés contre les attaques par fautes. La richesse mathématique des courbes elliptiques joue ici en faveur de l'attaquant.

Bibliographie

1. Steven Arno et Ferrell S. Wheeler. « Signed digit representations of minimal Hamming weight ». *IEEE Transactions on Computers*, 42(8) : 1007–1010, 1993.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, et Gene Tsudik. « A practical and provably secure coalition-resistant group signature scheme ». Dans M. Bellare, éditeur, *Advances in Cryptology – CRYPTO 2000*, volume 1880 de *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, 2000.
3. Giuseppe Ateniese, Marc Joye, et Gene Tsudik. « On the difficulty of coalition-resistance in group signature schemes ». Dans G. Persiano, éditeur, *2nd Workshop on Security in Communication Networks (SCN '99)*, Almafi, Italy, septembre 1999.
4. F. Bao, R.H. Deng, Y. Han, A. Jeng, A.D. Narasimbalu, et T. Ngair. « Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults ». Dans B. Christianson, B. Crispo, M. Lomas, et M. Roe, éditeurs, *Security Protocols*, volume 1361 de *Lecture Notes in Computer Science*, pages 115–124. Springer-Verlag, 1997.
5. Mihir Bellare, Anand Desai, David Pointcheval, et Phillip Rogaway. « Relations among notions of security for public-key encryption schemes ». Dans H. Krawczyk, éditeur, *Advances in Cryptology – CRYPTO '98*, volume 1462 de *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 1998.
6. Mihir Bellare et Phillip Rogaway. « Optimal asymmetric encryption ». Dans A. De Santis, éditeur, *Advances in Cryptology – EUROCRYPT '94*, volume 950 de *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.
7. Mihir Bellare et Phillip Rogaway. « The exact security of digital signatures - How to sign with RSA and Rabin ». Dans U. Maurer, éditeur, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 de *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.
8. « New threat model breaks crypto codes ». Bellcore's Press Release, septembre 1996. Archivé à l'URL http ://www.informatik.uni-mannheim.de/informatik/pi4/projects/Crypto/rgp/dfa/facts.html.
9. Olivier Benoît et Marc Joye. « Protecting RSA against fault attacks ». *Smart Times*, 4(2), 2002.
10. Ingrid Biehl, Bernd Meyer, et Volker Müller. « Differential fault attacks on elliptic curve cryptosystems ». Dans M. Bellare, éditeur, *Advances in Cryptology – CRYPT0 2000*, volume 1880 de *Lecture Notes in Computer Science*, pages 131–146. Springer-Verlag, 2000.

11. Eli Biham et Adi Shamir. « Differential fault analysis of secret key cryptosystems ». Dans B. Kaliski, éditeur, *Advances in Cryptology – CRYPTO '97*, volume 1294 de *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.
12. Olivier Billet et Marc Joye. « The Jacobi model of an elliptic curve and side-channel analysis ». Dans M. Fossorier, T. Høholdt, et A. Poli, éditeurs, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 2643 de *Lecture Notes in Computer Science*, pages 34–42. Springer-Verlag, 2003.
13. Simon R. Blackburn, Simon Blake-Wilson, Mike Burmester, et Steven D. Galbraith. « Weaknesses in shared RSA key generation protocols ». Dans M. Walker, éditeur, *Cryptography and Coding*, volume 1746 de *Lecture Notes in Computer Science*, pages 300–306. Springer-Verlag, 1999.
14. Daniel Bleichenbacher. « Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1 ». Dans H. Krawczyk, éditeur, *Advances in Cryptology – CRYPTO '98*, volume 1462 de *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1998.
15. Daniel Bleichenbacher, Wieb Bosma, et Arjen K. Lenstra. « Some remarks on Lucas-based cryptosystems ». Dans D. Coppersmith, éditeur, *Advances in Cryptology – CRYPTO '95*, volume 963 de *Lecture Notes in Computer Science*, pages 386–396. Springer-Verlag, 1995.
16. Daniel Bleichenbacher, Marc Joye, et Jean-Jacques Quisquater. « A new and optimal chosen-message attack on RSA-type cryptosystems ». Dans Y. Han, T. Okamoto, et S. Qing, éditeurs, *Information and Communications Security*, volume 1334 de *Lecture Notes in Computer Science*, pages 302–313. Springer-Verlag, 1997.
17. Daniel Bleichenbacher, Burt Kaliski, et Jessica Stadon. « Recent results on PKCS #1 : RSA encryption standard ». RSA Laboratories' Bulletin 7, RSA Laboratories, juin 20, 1998.
18. Dan Boneh. « Twenty years of attacks on the RSA cryptosystem ». *Notices of the AMS*, 46(2) : 203–213, 1999.
19. Dan Boneh, Richard A. DeMillo, et Richard J. Lipton. « On the importance of checking cryptographic protocols for faults ». Dans W. Fumy, éditeur, *Advances in Cryptology – EUROCRYPT '97*, volume 1233 de *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.
20. Dan Boneh, Richard A. DeMillo, et Richard J. Lipton. « On the importance of eliminating errors in cryptographic computations ». *Journal of Cryptology*, 14(2) : 101–119, 2001.
21. Dan Boneh et Matthew Franklin. « Efficient generation of shared RSA keys ». Dans B. Kaliski, éditeur, *Advances in Cryptology – CRYPTO '97*, volume 1294 de *Lecture Notes in Computer Science*, pages 425–439. Springer-Verlag, 1997.

22. Andrew D. Booth. « A signed binary multiplication technique ». *The Quaterly J. Mechanics and Applied Mathematics*, 4:236–240, 1951.

23. Jørgen Brandt, Ivan Damgård, et Peter Landrock. « Speeding up prime generation ». Dans H. Imai et R.L. Rivest, éditeurs, *Advances in Cryptology – ASIACRYPT '91*, volume 739 de *Lecture Notes in Computer Science*, pages 440–449. Springer-Verlag, 1993.

24. Éric Brier et Marc Joye. « Weierstraß elliptic curves and side-channel attacks ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 335–345. Springer-Verlag, 2002.

25. Éric Brier et Marc Joye. « Fast point multiplication on elliptic curves through isogenies ». Dans M. Fossorier, T. Høholdt, et A. Poli, éditeurs, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 2643 de *Lecture Notes in Computer Science*, pages 43–50. Springer-Verlag, 2003.

26. Jan Camenisch et Markus Stadler. « Efficient group signature schemes for large groups ». Dans B. Kaliski, éditeur, *Advances in Cryptology – CRYPTO '97*, volume 1294 de *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.

27. Sébastien Canard et Marc Girault. « Implementing group signatures with smart cards ». Dans P. Honeyman, éditeur, *Smart Card Research and Advanced Applications (CARDIS '02)*. Usenix Association, 2002.

28. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, et Pankaj Rohatgi. « Towards sound approaches to counteract power-analysis attacks ». Dans M. Wiener, éditeur, *Advances in Cryptology – CRYPTO '99*, volume 1666 de *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, 1999.

29. David Chaum et Eugène van Heyst. « Group signatures ». Dans D.W. Davies, éditeur, *Advances in Cryptology – EUROCRYPT '91*, volume 547 de *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.

30. Lidong Chen et Torben P. Pedersen. « New group signature schemes ». Dans A. De Santis, éditeur, *Advances in Cryptology – EUROCRYPT '94*, volume 950 de *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1995.

31. Benoît Chevallier-Mames, Mathieu Ciet, et Marc Joye. « Low-cost solutions for preventing simple side-channel analysis : Side-channel atomicity ». *IEEE Transactions on Computers*, À paraître.

32. Seng-Kiat Chua et San Ling. « A Rabin-type scheme on $y^2 \equiv x^3 + bx^2 \bmod n$ ». Dans T. Jiang et D.T. Lee, éditeurs, *Computing and Combinatorics (COCOON '97)*, volume 1276 de *Lecture Notes in Computer Science*, pages 186–191. Springer-Verlag, 1997.

33. D.V. Chudnovsky et G.V. Chudnovsky. « Sequences of numbers generated by addition in formal groups and new primality and factorization tests ». *Advances in Applied Mathematics*, 7 : 385–434, 1987.

34. Mathieu Ciet et Marc Joye. « Elliptic curve cryptosystems in the presence of permanent and transient faults ». *Designs, Codes and Cryptography*, À paraître.

35. W. Edwin Clark et J.J. Liang. « On arithmetic weight for a general radix representation of integers ». *IEEE Transactions on Information Theory*, IT-19 : 823–826, 1973.

36. Christophe Clavier et Marc Joye. « Universal exponentiation algorithm : A first step towards provable SPA-resistance ». Dans Ç.K. Koç, D. Naccache, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 de *Lecture Notes in Computer Science*, pages 300–308. Springer-Verlag, 2001.

37. Clifford Cocks. « Split knowledge of RSA parameters ». Dans M. Darnell, éditeur, *Cryptography and Coding*, volume 1355 de *Lecture Notes in Computer Science*, pages 89–95. Springer-Verlag, 1997.

38. Don Coppersmith. « Small solutions to polynomial equations, and low exponent RSA vulnerabilities ». *Journal of Cryptology*, 10(4) : 233–260, 1997.

39. Don Coppersmith, Matthew K. Franklin, Jacques Patarin, et Michael K. Reiter. « Low-exponent RSA with related messages ». Dans U. Maurer, éditeur, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 de *Lecture Notes in Computer Science*, pages 1–9. Springer-Verlag, 1996.

40. Don Coppersmith, Shai Halevi, et Charantli Jutla. « ISO 9796-1 and the new forgery strategy ». Présenté à la session impromptue de crypto '99, Santa Barbara, USA, août 1999.

41. Jean-Sébastien Coron. « Resistance against differential power analysis for elliptic curve cryptosystems ». Dans Ç.K. Koç et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems (CHES '99)*, volume 1717 de *Lecture Notes in Computer Science*, pages 292–302. Springer-Verlag, 1999.

42. Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, et Christophe Tymen. « GEM : A generic chosen-ciphertext secure encryption method ». Dans B. Preneel, éditeur, *Topics in Cryptology – CT-RSA 2002*, volume 2271 de *Lecture Notes in Computer Science*, pages 263–276. Springer-Verlag, 2002.

43. Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, et Christophe Tymen. « Optimal chosen-ciphertext secure encryption of arbitrary-length messages ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 17–33. Springer-Verlag, 2002.

44. Jean-Sébastien Coron, Marc Joye, David Naccache, et Pascal Paillier. « New attacks on PKCS #1 v1.5 encryption ». Dans B. Preneel, éditeur, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 de *Lecture Notes in Computer Science*, pages 369–381. Springer-Verlag, 2000.

45. Jean-Sébastien Coron, Marc Joye, David Naccache, et Pascal Paillier. « Universal padding schemes for RSA ». Dans M. Yung, éditeur, *Advances in Cryptology – CRYPTO 2002*, volume 2442 de *Lecture Notes in Computer Science*, pages 226–241. Springer-Verlag, 2002.

46. Jean-Sébastien Coron, David Naccache, et Julien P. Stern. « On the security of RSA padding ». Dans M. Wiener, éditeur, *Advances in Cryptology – CRYPTO '99*, volume 1666 de *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 1999.

47. Ronald Cramer et Victor Shoup. « A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack ». Dans H. Krawczyk, éditeur, *Advances in Cryptology – CRYPTO '98*, volume 1462 de *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.

48. George I. Davida. « Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem ». Rapport Technique TR-CS-82-2, University of Wisconsin, Milwaukee, USA, 1982.

49. Erik De Win, Serge Mister, Bart Preneel, et Michael Wiener. « On the performance of signature schemes based on elliptic curves ». Dans J.-P. Buhler, éditeur, *Algorithmic Number Theory Symposium*, volume 1423 de *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 1998.

50. N. Demytko. « A new elliptic curve based analogue of RSA ». Dans T. Helleseth, éditeur, *Advances in Cryptology – EUROCRYPT '93*, volume 765 de *Lecture Notes in Computer Science*, pages 40–49. Springer-Verlag, 1994.

51. Jean-François Dhem, Marc Joye, et Jean-Jacques Quisquater. « Normalisation in diminished-radix modulus transformation ». *Electronics Letters*, 33(23) : 1931, 1997.

52. Whitfield Diffie et Martin E. Hellman. « New directions in cryptography ». *IEEE Transactions on Information Theory*, IT-22(6) : 644–654, 1976.

53. Danny Dolev, Cynthia Dwork, et Moni Naor. « Non-malleable cryptography ». *SIAM Journal on Computing*, 30(2) : 391–347, 2000.

54. Josep Domingo-Ferrer, Antoni Martínez-Ballesté, et Francesc Sebé. « MICROCAST : Smart card based (micro) pay-per-view for multicast services ». Dans P. Honeyman, éditeur, *Smart Card Research and Advanced Applications (CARDIS '02)*, pages 125–134. Usenix Association, 2002.

55. Nathalie Feyt et Marc Joye. « A better use of smart cards in PKIs ». Dans *Gemplus Developer Conference*, Singapore, novembre 2002.

56. Nathalie Feyt, Marc Joye, David Naccache, et Pascal Paillier. « Off-line/on-line generation of RSA keys with smart cards ». Dans S.-P. Shieh, éditeur, *2nd International Workshop for Asian Public Key Infrastructures (IWAP 2002)*, pages 153–158, Taipei, Taiwan, octobre 2002.

57. Roger Fischlin. « Fast proof of plaintext-knowledge and deniable authentication based on Chinese Remainder Theorem ». Cryptology ePrint Archive Report 1999/006, IACR, 1999.

58. Yair Frankel, Philip D. MacKenzie, et Moti Yung. « Robust efficient distributed RSA-key generation ». Dans *Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 663–672. ACM Press, 1998.

59. Eiichiro Fujisaki et Tatsuaki Okamoto. « How to enhance the security of public-key encryption at minimum cost ». Dans H. Imai et Y. Zheng, éditeurs, *Public Key Cryptography*, volume 1560 de *Lecture Notes in Computer Science*, pages 53–68. Springer-Verlag, 1999.

60. Eiichiro Fujisaki et Tatsuaki Okamoto. « Secure integration of asymmetric and symmetric encryption schemes ». Dans M. Wiener, éditeur, *Advances in Cryptology – CRYPTO '99*, volume 1666 de *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.

61. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, et Jacques Stern. « RSA-OAEP is secure under the RSA assumption ». Dans J. Kilian, éditeur, *Advances in Cryptology – CRYPTO 2001*, volume 2139 de *Lecture Notes in Computer Science*, pages 260–274. Springer-Verlag, 2001.

62. Henri Gilbert, Dipankar Gupta, Andrew Odlyzko, et Jean-Jacques Quisquater. « Attacks on Shamir's 'RSA for paranoids' ». *Information Processing Letters*, 68 : 197–199, 1998.

63. Niv Gilboa. « Two party RSA key generation ». Dans M. Wiener, éditeur, *Advances in Cryptology – CRYPTO '99*, volume 1666 de *Lecture Notes in Computer Science*, pages 116–129. Springer-Verlag, 1999.

64. Aurore Gillet, Marc Joye, et Jean-Jacques Quisquater. « Cautionary note for protocols designers : Security proof is not enough ». Dans H. Orman et C. Meadows, éditeurs, *DIMACS Workshop on Design and Formal Verification of Security Protocols*, New-York, USA, septembre 1997. Disponible sur CDROM.

65. Daniel M. Gordon. « A survey of fast exponentiation methods ». *Journal of Algorithms*, 27 : 129–146, 1998.

66. François Grieu. « A chosen messages attack on the ISO/IEC 9796-1 signature scheme ». Dans B. Preneel, éditeur, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 de *Lecture Notes in Computer Science*, pages 70–80. Springer-Verlag, 2000.

67. L. Harn et H.Y. Lin. « An authenticated key agreement protocol without using one-way functions ». Dans *Eight National Conference on Information Security*, pages 155–160, Kaoshiung, mai 1998.

68. IEEE Std 1363-2000. *IEEE Standard Specifications for Public-Key Cryptography*. IEEE Computer Society, août 2000.
69. Tetsuya Izu et Tsuyoshi Takagi. « A fast parallel elliptic curve multiplication resistant against side channel attacks ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 280–296. Springer-Verlag, 2002.
70. Éliane Jaulmes et Antoine Joux. « A NICE cryptanalysis ». Dans B. Preneel, éditeur, *Advances in Cryptology – EUROCRYPT 2000*, volume 1880 de *Lecture Notes in Computer Science*, pages 382–391. Springer-Verlag, 2000.
71. Antoine Joux, Gwenaëlle Martinet, et Frédéric Valette. « Blockwise-adaptive attackers – Revisiting the (in)security of some provably secure encryption modes : CBC, GEM, IACBC ». Dans M. Yung, éditeur, *Advances in Cryptology – CRYPTO 2002*, volume 2442 de *Lecture Notes in Computer Science*, pages 17–30. Springer-Verlag, 2002.
72. Marc Joye. « Arithmétique algorithmique : Application au crypto-système à clé publique RSA ». Mémoire de fin d'études d'Ingénieur Civil en Mathématiques Appliquées, Université catholique de Louvain, Louvain-la-Neuve, janvier 1994.
73. Marc Joye. « *Security analysis of RSA-type cryptosystems* ». Thèse de doctorat en Sciences Appliquées, Université catholique de Louvain, Louvain-la-Neuve, octobre 1997.
74. Marc Joye. « Recovering lost efficiency of exponentiation algorithms on smart cards ». *Electronics Letters*, 38(19) : 1095–1097, 2002.
75. Marc Joye. « Cryptanalysis of a pay-as-you-watch system ». *Information Processing Letters*, 88(3) : 119–120, 2003.
76. Marc Joye. « Elliptic curves and side-channel analysis ». *ST Journal of System Research*, 4(1) : 17–21, 2003.
77. Marc Joye, Seungjoo Kim, et Narn-Yih Lee. « Cryptanalysis of two group signature schemes ». Dans M. Mambo et Y. Zheng, éditeurs, *Information Security*, volume 1729 de *Lecture Notes in Computer Science*, pages 271–275. Springer-Verlag, 1999.
78. Marc Joye, Narn-Yih Lee, et Tzonelih Hwang. « On the security of the Lee-Chang group signature scheme and its derivatives ». Dans M. Mambo et Y. Zheng, éditeurs, *Information Security*, volume 1729 de *Lecture Notes in Computer Science*, pages 47–51. Springer-Verlag, 1999.
79. Marc Joye, Arjen K. Lenstra, et Jean-Jacques Quisquater. « Chinese remaindering cryptosystems in the presence of faults ». *Journal of Cryptology*, 12(4) : 241–245, 1999.
80. Marc Joye et Pascal Paillier. « Constructive methods for the generation of prime numbers ». Dans S. Murphy, éditeur, *2nd NESSIE Workshop*, Egham, UK, septembre 2001.

81. Marc Joye et Pascal Paillier. « How to use RSA ; or how to improve the efficiency of RSA without loosing its security ». Dans *ISSE 2002*, Paris, France, octobre 2002.

82. Marc Joye, Pascal Paillier, et Serge Vaudenay. « Efficient generation of prime numbers ». Dans Ç.K. Koç et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 de *Lecture Notes in Computer Science*, pages 340–354. Springer-Verlag, 2000.

83. Marc Joye, Pascal Paillier, et Sung-Ming Yen. « Secure evaluation of modular functions ». Dans R.J. Hwang et C.K. Wu, éditeurs, *2001 International Workshop on Cryptology and Network Security*, pages 227–229, Taipei, Taiwan, septembre 2001.

84. Marc Joye et Richard Pinch. « Cheating in split-knowledge RSA parameter generation ». Dans D. Augot et C. Carlet, éditeurs, *Workshop on Coding and Cryptography*, pages 157–163, Paris, France, janvier 1999.

85. Marc Joye et Jean-Jacques Quisquater. « Efficient computation of full Lucas sequences ». *Electronics Letters*, 32(6) :537–538, 1996.

86. Marc Joye et Jean-Jacques Quisquater. « Cryptosystem of Chua and Ling ». *Electronics Letters*, 33(23) :1938, 1997.

87. Marc Joye et Jean-Jacques Quisquater. « On the importance of securing your bins : The garbage-man-in-the-middle attack ». Dans T. Matsumoto, éditeur, *4th ACM Conference on Computer and Communications Security*, pages 135–141. ACM Press, 1997.

88. Marc Joye et Jean-Jacques Quisquater. « Protocol failures for RSA-type cryptosystems using Lucas sequences and elliptic curves ». Dans M. Lomas, éditeur, *Security Protocols*, volume 1189 de *Lecture Notes in Computer Science*, pages 93–100. Springer-Verlag, 1997.

89. Marc Joye et Jean-Jacques Quisquater. Cryptanalysis of RSA-type cryptosystems : A visit. Dans R.N. Wright et P.G. Neumann, éditeurs, *Network Threats*, volume 38 de *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 21–31. American Mathematical Society, 1998.

90. Marc Joye et Jean-Jacques Quisquater. « Reducing the elliptic curve cryptosystem of Meyer-Müller to the cryptosystem of Rabin-Williams ». *Designs, Codes and Cryptography*, 14(1) :53–56, 1998.

91. Marc Joye et Jean-Jacques Quisquater. « Hessian elliptic curves and side-channel attacks ». Dans Ç.K. Koç, D. Naccache, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 de *Lecture Notes in Computer Science*, pages 402–410. Springer-Verlag, 2001.

92. Marc Joye et Jean-Jacques Quisquater. « On Rabin-type signatures ». Dans B. Honary, éditeur, *Cryptography and Coding*, volume 2260 de *Lecture Notes in Computer Science*, pages 99–113. Springer-Verlag, 2001.

93. Marc Joye, Jean-Jacques Quisquater, Feng Bao, et Robert H. Deng. « RSA-type signatures in the presence of transient faults ». Dans M. Darnell, éditeur, *Cryptography and Coding*, volume 1355 de *Lecture Notes in Computer Science*, pages 155–160. Springer-Verlag, 1997.

94. Marc Joye, Jean-Jacques Quisquater, et Tsuyoshi Takagi. « Are strong primes really stronger ? ». Dans D.J. Guan, éditeur, *Eight National Conference on Information Security*, pages 355–367, Kaoshung, Taiwan, mai 1998.

95. Marc Joye, Jean-Jacques Quisquater, et Tsuyoshi Takagi. « How to choose secret parameters for RSA and its extensions to elliptic curves ». *Designs, Codes and Cryptography*, 23(3) :297–316, 2001.

96. Marc Joye, Jean-Jacques Quisquater, Sung-Ming Yen, et Moti Yung. « Security paradoxes : How improving a cryptosystem may weaken it ». Dans *Ninth National Conference on Information Security*, pages 27–32, Taichung, Taiwan, mai 1999.

97. Marc Joye, Jean-Jacques Quisquater, Sung-Ming Yen, et Moti Yung. « Observability Analysis : Detecting when improved cryptosystems fail ». Dans B. Preneel, éditeur, *Topics in Cryptology – CT-RSA 2002*, volume 2271 de *Lecture Notes in Computer Science*, pages 17–29. Springer-Verlag, 2002.

98. Marc Joye, Jean-Jacques Quisquater, et Moti Yung. « On the power of misbehaving adversaries and security analysis of the original EPOC ». Dans D. Naccache, éditeur, *Topics in Cryptology – CT-RSA 2001*, volume 2020 de *Lecture Notes in Computer Science*, pages 208–222. Springer-Verlag, 2001.

99. Marc Joye et Christophe Tymen. « Compact encoding of non-adjacent forms with applications to elliptic curve cryptography ». Dans K. Kim, éditeur, *Public Key Cryptography*, volume 1992 de *Lecture Notes in Computer Science*, pages 353–364. Springer-Verlag, 2001.

100. Marc Joye et Christophe Tymen. « Protections against differential analysis for elliptic curve cryptography : An algebraic approach ». Dans Ç.K. Koç, D. Naccache, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 de *Lecture Notes in Computer Science*, pages 377–390. Springer-Verlag, 2001.

101. Marc Joye et Karine Villegas. « A protected division algorithm ». Dans P. Honeyman, éditeur, *Smart Card Research and Advanced Applications (CARDIS '02)*, pages 69–74. Usenix Association, 2002.

102. Marc Joye et Sung-Ming Yen. « Generation/Release of secrets using one-way cross-trees ». Dans T.L. Hwang et A.K. Lenstra, éditeurs, *Workshop on Cryptology and Information Security*, 1998 International Computer Symposium, pages 23–28, Tainan, Taiwan, décembre 1998.

103. Marc Joye et Sung-Ming Yen. « ID-based secret-key cryptography ». *ACM Operating Systems Review*, 32(4) :33–39, 1998.

104. Marc Joye et Sung-Ming Yen. « Optimal left-to-right binary signed-digit exponent recoding ». *IEEE Transactions on Computers*, 49(7) : 740–748, 2000.

105. Marc Joye et Sung-Ming Yen. « New minimal modified radix-r representation with applications to smart cards ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 375–384. Springer-Verlag, 2002.

106. Marc Joye et Sung-Ming Yen. « One-way cross-trees and their applications ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 346–356. Springer-Verlag, 2002.

107. Marc Joye et Sung-Ming Yen. « The Montgomery powering ladder ». Dans B.S. Kaliski Jr, Ç.K. Koç, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 de *Lecture Notes in Computer Science*, pages 291–302. Springer-Verlag, 2003.

108. Burton S. Kaliski, Jr. « A chosen message attack on Demytko's elliptic curve cryptosystem ». *Journal of Cryptology*, 10(1) : 71–72, 1997.

109. Seungjoo Kim, Junghee Cheon, Marc Joye, Seongan Lim, Masahiro Mambo, Dongho Won, et Yuliang Zheng. « Strong adaptive chosen-ciphertext attack with memory dump (or : The importance of the order of decryption and validation) ». Dans B. Honary, éditeur, *Cryptography and Coding*, volume 2260 de *Lecture Notes in Computer Science*, pages 114–127. Springer-Verlag, 2001.

110. Paul Kocher. « Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems ». Dans N. Koblitz, éditeur, *Advances in Cryptology – CRYPTO '96*, volume 1109 de *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

111. Paul Kocher, Joshua Jaffe, et Benjamin Jun. « Differential power analysis ». Dans M. Wiener, éditeur, *Advances in Cryptology – CRYPTO '99*, volume 1666 de *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

112. Kenji Koyama, Ueli M. Maurer, Tatsuaki Okamoto, et Scott A. Vanstone. « New public-key schemes based on elliptic curves over the ring $\mathbb{Z}_n$ ». Dans J. Feigenbaum, éditeur, *Advances in Cryptology – CRYPTO '91*, volume 576 de *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 1992.

113. Leslie Lamport. « Password authentication with insecure communication ». *Communications of the ACM*, 24(11) : 770–772, 1981.

114. Wei-Bin Lee et Chin-Chen Chang. « Efficient group signature scheme based on the discrete logarithm ». *IEE Proc.-Comput. Digit. Tech.*, 145(4) : 265–271, juillet 1998.

115. Pierre-Yvan Liardet et Nigel P. Smart. « Preventing SPA/DPA in ECC systems using the Jacobi form ». Dans Ç.K. Koç, D. Naccache,

et C. PAAR, éditeurs, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 de *Lecture Notes in Computer Science*, pages 391–401. Springer-Verlag, 2001.

116. R. LIDL, G.L. MULLEN, et G. TURNWALD. *Dickson Polynomials*. Longman Scientific & Technical, 1993.
117. Julio LÓPEZ et Ricardo DAHAB. « Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation ». Dans Ç.K. KOÇ et C. PAAR, éditeurs, *Cryptographic Hardware and Embedded Systems*, volume 1717 de *Lecture Notes in Computer Science*, pages 316–327. Springer-Verlag, 1999.
118. Anna LYSYANSKAYA et Zulfikar RAMZAN. « Group blind signatures : A scalable solution to electronic cash ». Dans R. HIRSCHFELD, éditeur, *Financial Cryptography (FC '98)*, volume 1465 de *Lecture Notes in Computer Science*, pages 184–197. Springer-Verlag, 1998.
119. John MALONE-LEE et Wenbo MAO. « Two birds one stone : Signcryption using RSA ». Dans M. JOYE, éditeur, *Topics in Cryptology – CT-RSA 2003*, volume 20 de *Lecture Notes in Computer Science*, pages 211–225. Springer-Verlag, 2003.
120. Alfred J. MENEZES, Paul C. VAN OORSCHOT, et Scott A. VANSTONE. *Handbook of Applied Cryptography*. CRC Press, 1997.
121. Bernd MEYER et Volker MÜLLER. « A public key cryptosystem based on elliptic curves over $\mathbb{Z}/n\mathbb{Z}$ equivalent to factoring ». Dans U. MAURER, éditeur, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 de *Lecture Notes in Computer Science*, pages 49–59. Springer-Verlag, 1996.
122. Peter L. MONTGOMERY. « Speeding the Pollard and elliptic curve methods of factorization ». *Mathematics of Computation*, 48(177) : 243–264, janvier 1987.
123. François MORAIN et Jorge OLIVOS. « Speeding up the computations on an elliptic curve using addition-subtraction chains ». Dans *Theoretical Informatics and Applications*, volume 24, pages 531–543, 1990.
124. Moni NAOR et Moti YUNG. « Public-key cryptosystems provably secure against chosen ciphertext attacks ». Dans *22nd ACM Annual Symposium on the Theory of Computing (STOC '90)*, pages 427–437. ACM Press, 1990.
125. Kenny Q. NGUYEN et Jacques TRAORÉ. « An online public auction protocol protecting bidder privacy ». Dans E. DAWSON, A. CLARK, et C. BOYD, éditeurs, *Information Security and Privacy*, volume 1841 de *Lecture Notes in Computer Science*, pages 427–442. Springer-Verlag, 2000.
126. Tatsuaki OKAMOTO et David POINTCHEVAL. « The GAP problems : A new class of problems for the security of cryptographic schemes ». Dans K. KIM, éditeur, *Public Key Cryptography*, volume 1992 de *Lecture Notes in Computer Science*, pages 104–118. Springer-Verlag, 2001.
127. Tatsuaki OKAMOTO et David POINTCHEVAL. « REACT : Rapid Enhanced-security Asymmetric Cryptosystem Transform ». Dans D. NACCACHE,

éditeur, *Topics in Cryptology – CT-RSA 2001*, volume 2020 de *Lecture Notes in Computer Science*, pages 159–175. Springer-Verlag, 2001.

128. Tatsuaki Okamoto et Shigenori Uchiyama. « A new public-key cryptosystem as secure as factoring ». Dans K. Nyberg, éditeur, *Advances in Cryptology – EUROCRYPT '98*, volume 1403 de *Lecture Notes in Computer Science*, pages 308–318. Springer-Verlag, 1998.
129. Tatsuaki Okamoto, Shigenori Uchiyama, et Eiichiro Fujisaki. « EPOC : Efficient probabilistic public-key encryption ». Submission to P1363a, novembre 1998.
130. Katsuyuki Okeya et Kouichi Sakurai. « Power analysis breaks elliptic curve cryptosystems even secure against the timing attack ». Dans B. Roy et E. Okamoto, éditeurs, *Progress in Cryptology – INDOCRYPT 2000*, volume 1977 de *Lecture Notes in Computer Science*, pages 178–190. Springer-Verlag, 2000.
131. Glenn Orton, Lloyd Peppard, et Stafford E. Tavares. « A design of a fast pipelined modular multiplier based on a diminished-radix algorithm ». *Journal of Cryptology*, 6(4) : 183–208, 1997.
132. David Pointcheval. « Chosen-ciphertext security for any one-way cryptosystem ». Dans H. Imai et Y. Zheng, éditeurs, *Public Key Cryptography*, volume 1751 de *Lecture Notes in Computer Science*, pages 129–146. Springer-Verlag, 2000.
133. Guillaume Poupard et Jacques Stern. « Generation of shared RSA keys by two parties ». Dans K. Ohta et D. Pei, éditeurs, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 de *Lecture Notes in Computer Science*, pages 11–24. Springer-Verlag, 1998.
134. Jean-Jacques Quisquater. « Fast modular exponentiation without division ». Présenté à la session impromptue de eurocrypt '90, Århus, Danemark, mai 1990.
135. Jean-Jacques Quisquater et Marc Joye. « Authentication of sequences with the SL_2 hash function : Application to video sequences ». *Journal of Computer Security*, 5(3) : 213–223, 1997.
136. Jean-Jacques Quisquater, Benoît Macq, Marc Joye, Nicolas Degand, et Alexis Bernard. « Practical solution to authentication of images with a secure camera ». Dans I.K. Sethi et R.C. Jain, éditeurs, *Storage and Retrieval for Image and Video Databases V*, volume 3022, pages 290–297. SPIE, 1997.
137. Michael O. Rabin. « Digitalized signatures and public-key functions as intractable as factorization ». Rapport Technique MIT/LCS/TR-212, Laboratory for Computer Science, MIT, janvier 1979.
138. Charles Rackoff et Daniel R. Simon. « Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack ». Dans J. Feigenbaum, éditeur, *Advances in Cryptology – CRYPTO '91*, volume 576, pages 433–444. Springer-Verlag, 1992.

139. G.W. Reitwiesner. « Binary arithmetic ». *Advances in Computers*, 1 : 201–308, 1960.
140. Ronald L. Rivest, Adi Shamir, et Leonard M. Adleman. « A method for obtaining digital signatures and public-key cryptosystems ». *Communications of the ACM*, 21(2) : 120–126, 1978.
141. RSA Laboratories. « PKCS #1 v2.1 : RSA Cryptography Standard », juin 2002.
142. Claus Peter Schnorr et Markus Jakobsson. « Security of signed ElGamal encryption ». Dans T. Okamoto, éditeur, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 de *Lecture Notes in Computer Science*, pages 73–89. Springer-Verlag, 2000.
143. Adi Shamir. « Identity-based cryptosystems and signature schemes ». Dans G.R. Blakley et D. Chaum, éditeurs, *Advances in Cryptology – Proceedings of CRYPTO 84*, volume 196 de *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985.
144. Adi Shamir. « RSA for paranoids ». *Cryptobytes*, 5(1) : 1–4, 1995.
145. Adi Shamir. « How to check modular exponentiations ». Présenté à la session impromptue d' eurocrypt '97, Constance, Allemagne, mai 1997.
146. Victor Shoup. « OAEP reconsidered ». Dans J. Kilian, éditeur, *Advances in Cryptology – CRYPTO 2001*, volume 2139 de *Lecture Notes in Computer Science*, pages 239–259. Springer-Verlag, 2001.
147. Peter J. Smith et Michael J.J. Lennon. « LUC : A new public key system ». Dans E.G. Douglas, éditeur, *Ninth IFIP Symposium on Computer Security*, pages 103–117. Elsevier Science Publishers, 1993.
148. Jacques Stern, David Pointcheval, John Malone-Lee, et Nigel P. Smart. « Flaws in applying proof methodologies to signature schemes ». Dans M. Yung, éditeur, *Advances in Cryptology – CRYPTO 2002*, volume 2442 de *Lecture Notes in Computer Science*, pages 93–110. Springer-Verlag, 2002.
149. Jean-Pierre Tillich et Gilles Zémor. « Hashing with SL_2 ». Dans Y. Desmedt, éditeur, *Advances in Cryptology – CRYPTO '94*, volume 839 de *Lecture Notes in Computer Science*, pages 40–49. Springer-Verlag, 1994.
150. Jacques Traoré. « Group signatures and their relevance to privacy-protecting off-line electronic cash systems ». Dans J. Pieprzyk, R. Safavi-Naini, et J. Seberry, éditeurs, *Information Security and Privacy*, volume 1587 de *Lecture Notes in Computer Science*, pages 228–243. Springer-Verlag, 1999.
151. Yuh-Min Tseng et Jinn-Ke Kan. « A novel ID-based group signature ». Dans T.L. Hwang et A.K. Lenstra, éditeurs, *Workshop on Cryptology and Information Security*, pages 159–164, Tainan, décembre 1998.
152. Yiannis Tsiounis et Moti Yung. « On the security of ElGamal based encryption ». Dans H. Imai et Y. Zheng, éditeurs, *Public Key Cryptography*, volume 1431 de *Lecture Notes in Computer Science*, pages 117–134. Springer-Verlag, 1998.

153. J.H. van Lint. *Introduction to Coding Theory*, volume 86 de *Graduate Texts in Mathematics*. Springer-Verlag, 1982.

154. Colin D. Walter. « Faster modular multiplication by operand scaling ». Dans J. Feigenbaum, éditeur, *Advances in Cryptology – CRYPTO '91*, volume 576 de *Lecture Notes in Computer Science*, pages 313–323. Springer-Verlag, 1992.

155. Colin D. Walter. « MIST : An efficient, randomized exponentiation algorithm for resisting power analysis ». Dans B. Preneel, éditeur, *Topics in Cryptology – CT-RSA 2002*, volume 2271 de *Lecture Notes in Computer Science*, pages 53–66. Springer-Verlag, 2001.

156. Sung-Ming Yen et Marc Joye. « Improved authenticated multiple-key agreement protocol ». *Electronics Letters*, 34(18) :1738–1739, 1998.

157. Sung-Ming Yen et Marc Joye. « Checking before output may not be enough against fault-based cryptanalysis ». *IEEE Transactions on Computers*, 49(9) :967–970, 2000.

158. Sung-Ming Yen et Chi-Sung Laih. « Fast algorithms for LUC digital signature computation ». *IEE Proc.-Comput. Digit Tech.*, 142(2) :165–169, mars 1995.

Annexe A

Quelques travaux

Cryptanalyse

A New and Optimal Chosen-Message Attack on RSA-type Cryptosystems 54

[Published in Y. Han, T. Okamoto, and S. Qing, eds, *Information and Communications Security*, vol. 1334 of *Lecture Notes in Computer Science*, pp. 302–313, Springer-Verlag, 1997.]

Daniel Bleichenbacher, Marc Joye, Jean-Jacques Quisquater

On the Importance of Securing Your Bins: The Garbage-Man-in-the-Middle Attack 66

[Published in T. Matsumoto, Ed., 4th ACM Conference on Computer and Communications Security, pp. 135–141, ACM Press, 1997.]

Marc Joye, Jean-Jacques Quisquater

Reducing the Elliptic Curve Cryptosystem of Meyer-Müller to the Cryptosystem of Rabin-Williams 80

[Published in *Designs, Codes and Cryptography* **14**(1):53–56, 1998.]

Marc Joye, Jean-Jacques Quisquater

New Attacks on PKCS #1 v1.5 Encryption 84

[Published in B. Preneel, Ed., *Advances in Cryptology – EUROCRYPT 2000*, vol. 1807 of *Lecture Notes in Computer Science*, pp. 369–381, Springer-Verlag, 2000.]

Jean-Sébastien Coron, Marc Joye, David Naccache, Pascal Paillier

On Rabin-type Signatures 97

[Published in B. Honary, Ed., *Cryptography and Coding*, vol. 2260 of *Lecture Notes in Computer Science*, pp. 99–113, Springer-Verlag, 2001.]

Marc Joye, Jean-Jacques Quisquater

On the Power of Misbehaving Adversaries and Security Analysis of the Original EPOC 112

[Published in D. Naccache, Ed., *Topics in Cryptology – CT-RSA 2001*, vol. 2020 of *Lecture Notes in Computer Science*, pp. 208–222, Springer-Verlag, 2001.]

Marc Joye, Jean-Jacques Quisquater, Moti Yung

Nouveaux schémas cryptographiques

A Practical and Provably Secure Coalition-Resistant Group Signature Scheme 127

[Published in M. Bellare, Ed., *Advances in Cryptology – CRYPTO 2000*, vol. 1880 of *Lecture Notes in Computer Science*, pp. 255–270, Springer-Verlag, 2000.]

Giuseppe Ateniese, Jan Camenisch, Marc Joye, Gene Tsudik

Algorithmes efficaces

Implantations sécurisées

A New and Optimal Chosen-Message Attack on RSA-type Cryptosystems

[Published in Y. Han, T. Okamoto, and S. Qing, eds, *Information and Communications Security*, vol. 1334 of *Lecture Notes in Computer Science*, pp. 302–313, Springer-Verlag, 1997.]

Daniel Bleichenbacher[1], Marc Joye[2], and Jean-Jacques Quisquater[3]

[1] Bell Laboratories
700 Mountain Av. , Murray Hill, NJ 07974, U.S.A.
E-mail: bleichen@research.bell-labs.com
[2] UCL Crypto Group, Dép. de Mathématique, Université de Louvain
Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium
E-mail: joye@agel.ucl.ac.be
[3] UCL Crypto Group, Lab. de Microélectronique, Université de Louvain
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium
E-mail: jjq@dice.ucl.ac.be

Abstract. Chosen-message attack on RSA is usually considered as an inherent property of its homomorphic structure. In this paper, we show that non-homomorphic RSA-type cryptosystems are also susceptible to a chosen-message attack. In particular, we prove that only *one* message is needed to mount a successful chosen-message attack against the Lucas-based systems and Demytko's elliptic curve system.

Keywords. Chosen-message attack, signature forgery, RSA, Lucas-based systems, Demytko's elliptic curve system.

1 Introduction

The most used public-key cryptosystem is certainly the RSA [12]. Due to its popularity, the RSA was subject to an extensive cryptanalysis. Many attacks are based on the multiplicative nature of RSA [5]. To overcome this vulnerability, numerous generalizations of the original RSA were proposed and broken.

Later, other structures were envisaged to implement analogues of RSA. This seemed to be the right way to foil the homomorphic attacks. So, a cryptosystem based on Lucas sequences was proposed in [10] and analyzed in [11] by Müller and Nöbauer. The authors use Dickson polynomials to describe their scheme; however, Dickson polynomials can be rephrased in terms of Lucas sequences [2, 14]. The Lucas sequences play the same role in this scheme as exponentiations in RSA.

In 1985, Koblitz and Miller independently suggested the use of elliptic curves in cryptography [7, 9]. Afterwards, Koyama *et al.* [8] and Demytko [4] exhibited new one-way trapdoor functions on elliptic curves in order to produce analogues of RSA. Demytko's system has the particularity to only use the first coordinate and is therefore not subject to the chosen-message attack described in [8].

The Lucas-based cryptosystems and Demytko's elliptic curve cryptosystem seem to be resistant against homomorphic attack. However, the existence of a chosen-message forgery that needs two messages has been described in [1]. Kaliski found a similar attack on Demytko's system [6].

In this paper, we describe a new chosen-message attack which needs only one message. This new attack shows that the RSA-type cryptosystems are even closer related to RSA, i.e. it shows that all the attacks based on the multiplicative nature of the original RSA can straightforward be adapted to any RSA-type cryptosystem. We illustrate this topic with the common modulus failure [13].

The remainder of this paper is organized as follows. In Section 2, we review the Lucas-based and Demytko's elliptic curve cryptosystems. The reader who is not not familiar with these systems may first read the appendix. We present our attack in Section 3 and apply it in Section 4. In Section 5, we revisit the common modulus failure. Finally, we conclude in Section 6.

2 RSA-type Cryptosystems

In this section, we present cryptosystems based on Lucas sequences [10, 11, 14] and on elliptic curves [4]. We only outline the systems, for a detailed description we refer to the original papers.

2.1 Lucas-based RSA

The Lucas-based scheme can briefly be described as follows. Each user A chooses two large primes p and q and an exponent e that is relatively prime to $(p^2 - 1)(q^2 - 1)$, computes $n = pq$, and publishes n and e as his public key. The corresponding $d \equiv e^{-1} \pmod{\operatorname{lcm}(p-1, p+1, q-1, q+1)}$ is kept secret.

A's public parameters: n and e.
A's secret parameters: p, q and d.

A message m is encrypted by computing $c \equiv v_e(m, 1) \pmod{n}$. It is decrypted using the secret key d by $m \equiv v_d(c, 1) \pmod{n}$. The correctness of this system is based on Proposition 3 (in appendix) as $v_d\big(v_e(m, 1), 1\big) \equiv v_{de}(m, 1) \equiv v_1(m, 1) \equiv m \pmod{n}$. Signatures are generated accordingly by exchanging the roles of the public and secret parameters e and d.

2.2 Demytko's system

Similarly to RSA, to setup Demytko's system, each user A chooses two large primes p and q, and publishes their product $n = pq$. He publicly selects integers

a and b such that $\gcd(4a^3 + 27b^2, n) = 1$. Then once and for all, he computes

$$N_n = \operatorname{lcm}\left(\#E_p(a,b), \#E_q(a,b), \#\overline{E_p(a,b)}, \#\overline{E_q(a,b)}\right). \tag{1}$$

He randomly chooses the public encryption key e such that $\gcd(e, N_n) = 1$, and computes the secret decryption key d according to $ed \equiv 1 \bmod N_n$.

A's public parameters: n, a, b and e.
A's secret parameters: p, q, N_n and d.

It is useful to introduce some notation. The $x-$ and the $y-$coordinates of a point $\mathbf{P}$ will respectively be denoted by $x(\mathbf{P})$ and $y(\mathbf{P})$. To send a message m to Alice, Bob uses Alice's public key e and computes the corresponding ciphertext $c \equiv x(e\mathbf{M}) \pmod n$ where $\mathbf{M}$ is a point having its $x-$coordinate equal to m. Note that, from Proposition 1, the computation of $c \equiv x(e\mathbf{M}) \pmod n$ does not require the knowledge of $y(\mathbf{M})$.

Using her secret key d, Alice can recover the plaintext m by computing $m \equiv x(d\mathbf{C}) \pmod n$ where $\mathbf{C}$ is a point having its $x-$coordinate equal to c. Note also that Alice has not to know $y(\mathbf{C})$.

Remark 1. To speed up the computations, Alice can choose $p, q \equiv 2 \bmod 3$ and $a = 0$. In that case, $N_n = \operatorname{lcm}(p+1, q+1)$. The same conclusion holds by choosing $p, q \equiv 3 \bmod 4$ and $b = 0$ (see [8]).

Remark 2. For efficiency reasons, it is also possible to define a message-dependent system (see [4]).

3 Sketch of the New Attack

Let $n = pq$ be a RSA modulus. Let e and d be respectively the public key and the secret key of Alice, according to $ed \equiv 1 \pmod{\Phi(n)}$. The public key e is used to encrypt messages and verify signatures; the secret key d is used to decrypt ciphertexts and to sign messages.

Suppose a cryptanalyst (say Carol) wants to make Alice to sign message m without her consent. Carol can proceeds as follows. She chooses a random number k and asks Alice to sign (or to decrypt) $m' \equiv mk^e \pmod n$. Carol gets then $c' \equiv m^d (k^e)^d \equiv m^d k$, and therefore the signature c of message m as $c \equiv c' k^{-1} \pmod n$.

Consequently, chosen-message attacks against RSA seem quite naturally to be a consequence of its multiplicative structure. By reformulating this attack with the extended Euclidean algorithm, it appears that non-homomorphic cryptosystems are also susceptible to a chosen-message attack. Applying to RSA, the attack goes as follows.

Input: A message m and the public key n, e of Alice.

Step 1: Carol chooses an integer k relatively prime to e. Then she uses the extended Euclidean algorithm to find $r, s \in \mathbb{Z}$ such that $kr + es = 1$.
Step 2: Carol computes $m' \equiv m^k \pmod{n}$.
Step 3: Next, she asks Alice to sign m' and gets therefore

$$c' \equiv {m'}^d \pmod{n}.$$

Step 4: Consequently, Carol can compute the signature c of m by

$$c \equiv {c'}^r m^s \pmod{n}. \tag{2}$$

Output: The signature c of message m.

Proof. From $kr+es = 1$, it follows $d = d(kr+es) \equiv dkr+s \pmod{\Phi(n)}$. Hence, $c \equiv m^d \equiv m^{dkr} m^s \equiv \left(m^{dk}\right)^r m^s \equiv {c'}^r m^s \pmod{n}$. □

Remark 3. This attack can also be considered as a generalization of the Davida's attack [3].

4 Applications

The previous attack applies also to non-homomorphic cryptosystems. In this section, we show how it works against Lucas-based systems and Demytko's system.

4.1 Attacking Lucas-based systems

The cryptanalyst Carol can try to get a signature c on a message m in the following way.

Input: A message m and the public key n, e of Alice.
Step 1: Carol chooses an integer k relatively prime to e. Then she uses extended Euclidean algorithm to find $r, s \in \mathbb{Z}$ such that $kr + es = 1$.
Step 2: Next she computes $m' \equiv v_k(m, 1) \pmod{n}$.
Step 3: Now she asks Alice to sign m'. If Alice does so then Carol knows c' such that

$$c' \equiv v_d(m', 1) \pmod{n}.$$

Step 4: Finally Carol computes the signature c of m as follows

$$v_{rkd}(m, 1) \equiv v_r(c', 1) \pmod{n}, \tag{3}$$

$$u_{rkd}(m, 1) \equiv \frac{u_k(m, 1)u_r(c', 1)}{u_e(c', 1)} \pmod{n}, \tag{4}$$

$$c = v_d(m, 1) \equiv \frac{v_{rkd}(m, 1)v_s(m, 1)}{2} + \frac{\Delta\, u_{rkd}(m, 1)u_s(m, 1)}{2} \pmod{n} \tag{5}$$

where $\Delta = m^2 - 4$.

Output: The signature c of message m.

Proof. Equation (3) follows from (13)[1] since

$$v_r(c',1) \equiv v_r\big(v_{kd}(m,1),1\big) \equiv v_{rkd}(m,1) \pmod{n}.$$

Equation (4) is a consequence of (14) and

$$\begin{aligned} u_{rkd}(m,1)u_e(c',1) &\equiv u_r\big(v_{kd}(m,1),1\big)u_{kd}(m,1)u_e\big(v_{kd}(m,1),1\big) \\ &\equiv u_r\big(v_{kd}(m,1),1\big)u_{kde}(m,1) \\ &\equiv u_r\big(v_{kd}(m,1),1\big)u_k(m,1) \pmod{n}. \end{aligned}$$

Moreover, $kr + es = 1$ implies $v_d(m,1) = v_{rkd+des}(m,1) = v_{rkd+s}(m,1)$. Hence Equation (5) is an application of (15). □

Remark 4. This attack is the analogue to the chosen-message attack on RSA presented in Section 3, by using algebraic numbers (replace m by $\alpha = (m + \sqrt{\Delta})/2$ and use Equation (9)). The only additional step to be proved is that $u_{kd}(m,1)$ is computable from m and $v_{kd}(m,1)$. This can be shown by using (14) and noting that

$$u_k(m,1) \equiv u_{kde}(m,1) \equiv u_{kd}(m,1)u_e\big(v_{kd}(m,1),1\big) \pmod{n}.$$

If $\alpha = m/2 + \sqrt{\Delta}/2$ then the signature $v_{kd}(m,1)$ on the message $v_k(m,1)$ can be used to compute

$$\alpha^{kd} \equiv v_{kd}(m,1)/2 + u_{kd}(m,1)\sqrt{\Delta}/2.$$

Once α^{kd} is known, $\alpha^d = v_d(m,1)/2 + u_d(m,1)\sqrt{\Delta}/2$ can be computed from

$$\alpha^d \equiv \alpha^{(kr+es)d} \equiv \big(\alpha^{kd}\big)^r \alpha^s \pmod{n}.$$

Hence (3) and (4) correspond to the computation of c'^r and (5) corresponds to the multiplication of c'^r by m^s in (2).

4.2 Attacking Demytko's system

Before showing that a similar attack applies to Demytko's system, we need to prove the following proposition.

Proposition 1. *Let p be a prime greater than 3, and let $E_p(a,b)$ be an elliptic curve over $\mathbb{Z}_p$. If $\mathbf{P} \in E_p(a,b)$ or if $\mathbf{P} \in \overline{E_p(a,b)}$, then the computations of $x(k\mathbf{P})$ and $\dfrac{y(k\mathbf{P})}{y(\mathbf{P})}$ depend only on $x(\mathbf{P})$.*

[1] (9) to (22) refer to equations in the appendix.

Proof. Letting $X_j := x(j\mathbf{P})$ and $Y_j := \frac{y(j\mathbf{P})}{y(\mathbf{P})}$, the tangent-and-chord composition rule on elliptic curves gives the following formulas

$$\begin{aligned}
X_{2j} = x(j\mathbf{P}+j\mathbf{P}) &= \begin{cases} \left(\frac{3\,x(j\mathbf{P})^2+a}{2\,y(j\mathbf{P})}\right)^2 - 2\,x(j\mathbf{P}) & \text{if } \mathbf{P} \in E_p(a,b) \\ \left(\frac{3\,x(j\mathbf{P})^2+a}{2D_p\,y(j\mathbf{P})}\right)^2 D_p - 2\,x(j\mathbf{P}) & \text{if } \mathbf{P} \in \overline{E_p(a,b)} \end{cases} \\
&= \tfrac{1}{X_1^3+aX_1+b}\left(\tfrac{3X_j^2+a}{2Y_j}\right)^2 - 2X_j, \\
Y_{2j} = \tfrac{y(j\mathbf{P}+j\mathbf{P})}{y(\mathbf{P})} &= \begin{cases} \frac{\left(\frac{3\,x(j\mathbf{P})^2+a}{2\,y(j\mathbf{P})}\right)\left(x(j\mathbf{P})-x(2j\mathbf{P})\right)-y(j\mathbf{P})}{y(\mathbf{P})} & \text{if } \mathbf{P} \in E_p(a,b) \\ \frac{\left(\frac{3\,x(j\mathbf{P})^2+a}{2D_p\,y(j\mathbf{P})}\right)\left(x(j\mathbf{P})-x(2j\mathbf{P})\right)-y(j\mathbf{P})}{y(\mathbf{P})} & \text{if } \mathbf{P} \in \overline{E_p(a,b)} \end{cases} \\
&= \tfrac{1}{X_1^3+aX_1+b}\left(\tfrac{3X_j^2+a}{2Y_j}\right)(X_j - X_{2j}) - Y_j, \\
X_{2j+1} &= x\big(j\mathbf{P}+(j+1)\mathbf{P}\big) \\
&= \begin{cases} \left(\frac{y(j\mathbf{P})-y((j+1)\mathbf{P})}{x(j\mathbf{P})-x((j+1)\mathbf{P})}\right)^2 - x(j\mathbf{P}) - x\big((j+1)\mathbf{P}\big) & \text{if } \mathbf{P} \in E_p(a,b) \\ \left(\frac{y(j\mathbf{P})-y((j+1)\mathbf{P})}{x(j\mathbf{P})-x((j+1)\mathbf{P})}\right)^2 D_p - x(j\mathbf{P}) - x\big((j+1)\mathbf{P}\big) & \text{if } \mathbf{P} \in \overline{E_p(a,b)} \end{cases} \\
&= (X_1^3+aX_1+b)\left(\tfrac{Y_j-Y_{j+1}}{X_j-X_{j+1}}\right)^2 - X_j - X_{j+1}, \\
Y_{2j+1} &= \tfrac{y\left(j\mathbf{P}+(j+1)\mathbf{P}\right)}{y(\mathbf{P})} \\
&= \tfrac{\left(\frac{y(j\mathbf{P})-y((j+1)\mathbf{P})}{x(j\mathbf{P})-x((j+1)\mathbf{P})}\right)\left(x(j\mathbf{P})-x((2j+1)\mathbf{P})\right)-y(j\mathbf{P})}{y(\mathbf{P})} \quad \text{if } \mathbf{P} \in E_p(a,b) \text{ or } \overline{E_p(a,b)} \\
&= \tfrac{Y_j-Y_{j+1}}{X_j-X_{j+1}}(X_j - X_{2j+1}) - Y_j.
\end{aligned}$$

So X_k and Y_k can be computed from $X_1 = x(\mathbf{P})$ and $Y_1 = 1$ by using the binary method. □

Then, the message forgery goes as follows.

Input: A message m and the public key n, e of Alice.
Note that m is the $x-$coordinate of a point $\mathbf{M}$, i.e. $m = x(\mathbf{M})$.

Step 1: The cryptanalyst Carol chooses a random k relatively prime to e. Then she uses extended Euclidean algorithm to find $r, s \in \mathbb{Z}$ such that $kr + es = 1$.

Step 2: From $x(\mathbf{M})$, Carol computes $m' = x(\mathbf{M}') \equiv x(k\mathbf{M}) \pmod{n}$. Next, she asks Alice to sign m'. So, Carol obtains the signature

$$c' = x(\mathbf{C}') \equiv x(d\mathbf{M}') \pmod{n}.$$

Step 3: Finally, Carol finds the signature $c = x(\mathbf{C}) \equiv x(d\mathbf{M}) \pmod{n}$ of message m as follows.

3a) If $x(r\mathbf{C}') \not\equiv x(s\mathbf{M}) \pmod{n}$ then, using Proposition 1, Carol can compute

$$\frac{y(k\mathbf{M})}{y(\mathbf{M})}, \frac{y(r\mathbf{C}')}{y(\mathbf{C}')} \text{ and } \frac{y(e\mathbf{C}')}{y(\mathbf{C}')} \tag{6}$$

and

$$c \equiv (m^3 + am + b) \left[\frac{\frac{y(k\mathbf{M})}{y(\mathbf{M})} \frac{y(r\mathbf{C}')}{y(\mathbf{C}')} \left(\frac{y(e\mathbf{C}')}{y(\mathbf{C}')} \right)^{-1} - \frac{y(s\mathbf{M})}{y(\mathbf{M})}}{x(r\mathbf{C}') - x(s\mathbf{M})} \right]^2 - x(r\mathbf{C}') - x(s\mathbf{M}) \pmod{n}. \tag{7}$$

3b) Otherwise, the signature is given by

$$c \equiv \frac{\big[3\,x(r\mathbf{C}')^2 + a\big]^2}{4\big[x(r\mathbf{C}')^3 + a\,x(r\mathbf{C}') + b\big]} - 2\,x(r\mathbf{C}') \pmod{n}. \tag{8}$$

Output: The signature c of message m.

Proof. Since $kr + es = 1$, $d \equiv krd + esd \equiv krd + s \pmod{N_n}$. So,

$$x(\mathbf{C}) \equiv x(d\mathbf{M}) \equiv x\big([krd + s]\mathbf{M}\big) \equiv x(r\mathbf{C}' + s\mathbf{M}) \pmod{n}.$$

a) If $x(r\mathbf{C}') \not\equiv x(s\mathbf{M}) \pmod{n}$, then

$$\begin{aligned} &x(r\mathbf{C}' + s\mathbf{M}) \\ &\quad\equiv \left(\frac{y(r\mathbf{C}') - y(s\mathbf{M})}{x(r\mathbf{C}') - x(s\mathbf{M})} \right)^2 - x(r\mathbf{C}') - x(s\mathbf{M}) \\ &\quad\equiv y(\mathbf{M})^2 \left[\frac{\frac{y(k\mathbf{M})}{y(\mathbf{M})} \frac{y(r\mathbf{C}')}{y(\mathbf{C}')} \frac{y(\mathbf{C}')}{y(e\mathbf{C}')} - \frac{y(s\mathbf{M})}{y(\mathbf{M})}}{x(r\mathbf{C}') - x(s\mathbf{M})} \right]^2 - x(r\mathbf{C}') - x(s\mathbf{M}) \pmod{n} \end{aligned}$$

since

$$\frac{y(r\mathbf{C}')}{y(\mathbf{M})} = \frac{y(r\mathbf{C}')}{y(\mathbf{C}')} \frac{y(\mathbf{C}')}{y(k\mathbf{M})} \frac{y(k\mathbf{M})}{y(\mathbf{M})}$$

and $y(k\mathbf{M}) \equiv y(edk\mathbf{M}) \equiv y(e\mathbf{C}') \pmod{n}$.

b) Otherwise, since $\gcd(d, N_n) = 1$ it follows that $r\mathbf{C}' \not\equiv -s\mathbf{M} \pmod{n}$ and therefore

$$x(r\mathbf{C}' + s\mathbf{M}) \equiv \left(\frac{3\,x(rC')^2 + a}{2\,y(r\mathbf{C}')} \right)^2 - x(r\mathbf{C}') - x(s\mathbf{M}) \pmod{n}.$$

□

5 Common Modulus Attack

Simmons pointed out in [13] that the use of a common RSA modulus is dangerous. Indeed, if a message is sent to two users that have coprime public encryption keys, then the message can be recovered.

Because our chosen-message attack requires only one message, the Lucas-based systems and Demytko's elliptic curve system are vulnerable to the common modulus attack. We shall illustrate this topic on Demyko's system.

Let (e_1, d_1) and (e_2, d_2) be two pairs of encryption/decryption keys and let $m = x(\mathbf{M})$ be the message being encrypted. Assuming e_1 and e_2 are relatively prime, the cryptanalyst Carol can recover m from the ciphertexts $c_1 = x(\mathbf{C_1}) \equiv x(e_1\mathbf{M}) \pmod{n}$ and $c_2 = x(\mathbf{C_2}) \equiv x(e_2\mathbf{M}) \pmod{n}$ as follows.

Carol uses the extended Euclidean algorithm to find integers r and s such that $re_1 + se_2 = 1$. Then, she computes $x(\mathbf{M}) = x((re_1+se_2)\mathbf{M}) \equiv x(r\mathbf{C_1}+s\mathbf{C_2}) \pmod{n}$ as follows. If $x(r\mathbf{C_1}) \not\equiv x(s\mathbf{C_2}) \pmod{n}$, then

$$m \equiv (c_1^3 + ac_1 + b) \left[\frac{\frac{y(r\mathbf{C_1})}{y(\mathbf{C_1})} - \frac{y(e_2\mathbf{C_1})}{y(\mathbf{C_1})} \frac{y(s\mathbf{C_2})}{y(\mathbf{C_2})} \left(\frac{y(e_1\mathbf{C_2})}{y(\mathbf{C_2})} \right)^{-1}}{x(r\mathbf{C_1}) - x(s\mathbf{C_2})} \right]^2 - x(r\mathbf{C_1}) - x(s\mathbf{C_2}) \pmod{n}$$

otherwise

$$m \equiv \frac{\left[3\,x(r\mathbf{C_1})^2 + a\right]^2}{4\left[x(r\mathbf{C_1})^3 + a\,x(r\mathbf{C_1}) + b\right]} - 2\,x(r\mathbf{C_1}) \pmod{n}.$$

Proof. Straightforward since $y(e_2\mathbf{C_1}) \equiv y(e_1\mathbf{C_2}) \pmod{n}$. □

6 Conclusion

We have presented a new type of chosen-message attack. Our formulation has permitted to mount a successful chosen-message attack with only one message against Lucas-based systems and Demytko's system. This also proved that the use of non-homomorphic systems is not necessarily the best way to foil chosen-message attacks.

Acknowledgments

The second author is grateful to Victor Miller for providing some useful comments to enhance the presentation of the paper.

References

1. D. Bleichenbacher, W. Bosma, and A. K. Lenstra. Some remarks on Lucas-based cryptosystems. In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 386–396. Springer-Verlag, 1995.
2. D. M. Bressoud. *Factorization and primality testing*. Undergraduate Texts in Mathematics. Springer-Verlag, 1989.
3. G. Davida. Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem. Technical Report TR-CS-82-2, Dept. of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, USA, October 1982.
4. N. Demytko. A new elliptic curve based analogue of RSA. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 40–49. Springer-Verlag, 1994.
5. D. E. Denning. Digital signatures with RSA and other public-key cryptosystems. *Communications of the ACM*, 27(4):388–392, April 1984.
6. B. S. Kaliski Jr. A chosen message attack on Demytko's elliptic curve cryptosystem. *Journal of Cryptology*, 10(1):71–72, 1997.
7. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
8. K. Koyama, U. M. Maurer, T. Okamoto, and S. A. Vanstone. New public-key schemes based on elliptic curves over the ring $\mathbb{Z}_n$. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 1991.
9. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer-Verlag, 1986.
10. W. B. Müller and R. Nöbauer. Some remarks on public-key cryptosystems. *Sci. Math. Hungar*, 16:71–76, 1981.
11. W. B. Müller and R. Nöbauer. Cryptanalysis of the Dickson scheme. In J. Pichler, editor, *Advances in Cryptology – EUROCRYPT '85*, volume 219 of *Lecture Notes in Computer Science*, pages 50–61. Springer-Verlag, 1986.
12. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
13. G. J. Simmons. A weak privacy protocol using the RSA cryptoalgorithm. *Cryptologia*, 7:180–182, 1983.
14. P. J. Smith and M. J. J. Lennon. LUC: A new public key system. In E. G. Douglas, editor, *Ninth IFIP Symposium on Computer Security*, pages 103–117. Elsevier Science Publishers, 1993.

A Basic Facts

A.1 Lucas sequences

Let P, Q be integers, $\Delta = P^2 - 4Q$ be a non-square, $\alpha = \frac{P+\sqrt{\Delta}}{2}$ and $\beta = \overline{\alpha} = \frac{P-\sqrt{\Delta}}{2}$ be the roots of $x^2 - Px + Q = 0$ in the quadratic field $\mathbb{Q}(\sqrt{\Delta})$. The

Lucas sequences $v_k(P,Q)$ and $u_k(P,Q)$ for $k \in \mathbb{Z}$ are then defined as the integers satisfying

$$\alpha^k := \frac{v_k(P,Q)}{2} + \frac{u_k(P,Q)\sqrt{\Delta}}{2}. \tag{9}$$

From $\alpha^2 = P\alpha - Q$ follows $\alpha^k = P\alpha^{k-1} - Q\alpha^{k-2}$. Hence the Lucas sequences satisfy the following recurrence relation

$$\begin{aligned} &v_0(P,Q) = 2; \quad v_1(P,Q) = P; \quad v_k(P,Q) = Pv_{k-1}(P,Q) - Qv_{k-2}(P,Q), \\ &u_0(P,Q) = 0; \quad u_1(P,Q) = 1; \quad u_k(P,Q) = Pu_{k-1}(P,Q) - Qu_{k-2}(P,Q). \end{aligned}$$

This recurrence relation is sometimes used as an alternative definition of Lucas sequences. Since conjugation and exponentiation are exchangeable it follows

$$\beta^k = \overline{\alpha^k} = \frac{v_k(P,Q)}{2} - \frac{u_k(P,Q)\sqrt{\Delta}}{2}.$$

From this equation and from (9) it follows that

$$v_k(P,Q) = \alpha^k + \beta^k, \tag{10}$$

$$\text{and} \quad u_k(P,Q) = \frac{\alpha^k - \beta^k}{\alpha - \beta}. \tag{11}$$

The next proposition states some well-known properties of Lucas sequences.

Proposition 2.

$$4Q^k = v_k(P,Q)^2 - \Delta u_k(P,Q)^2 \tag{12}$$

$$v_{km}(P,Q) = v_k\big(v_m(P,Q), Q^m\big) \tag{13}$$

$$u_{km}(P,Q) = u_m(P,Q)u_k\big(v_m(P,Q), Q^m\big) \tag{14}$$

$$v_{k+m}(P,Q) = \frac{v_k(P,Q)v_m(P,Q)}{2} + \frac{\Delta u_k(P,Q)u_m(P,Q)}{2} \tag{15}$$

$$u_{k+m}(P,Q) = \frac{u_k(P,Q)v_m(P,Q)}{2} + \frac{v_k(P,Q)u_m(P,Q)}{2} \tag{16}$$

Proof. Equation (12) can be proved as follows.

$$\begin{aligned} 4Q^k &= 4(\alpha\overline{\alpha})^k = 2\alpha^k 2\overline{\alpha^k} \\ &= (v_k(P,Q) + u_k(P,Q)\sqrt{\Delta})(v_k(P,Q) - u_k(P,Q)\sqrt{\Delta}) \\ &= v_k(P,Q)^2 - \Delta u_k(P,Q)^2. \end{aligned}$$

Equation (12) now implies that

$$\begin{aligned} \alpha^k &= \frac{v_k(P,Q)}{2} + \frac{u_k(P,Q)\sqrt{\Delta}}{2} = \frac{v_k(P,Q)}{2} + \frac{\sqrt{u_k(P,Q)^2\Delta}}{2} \\ &= \frac{v_k(P,Q)}{2} + \frac{\sqrt{v_k(P,Q)^2 - 4Q^k}}{2} \end{aligned}$$

and hence $\alpha^k = P'/2 + \sqrt{P'^2 - 4Q'}/2$ with $P' = v_k(P,Q)$ and $Q' = Q^k$. Thus we have

$$\begin{aligned}(\alpha^k)^m &= \frac{v_m(P',Q')}{2} + \frac{u_m(P',Q')\sqrt{P'^2 - 4Q'}}{2} \\ &= \frac{v_m(P',Q')}{2} + \frac{u_m(P',Q')u_k(P,Q)\sqrt{\Delta}}{2}.\end{aligned}$$

Comparing the coefficients of this equation with

$$\alpha^{km} = v_{km}(P,Q)/2 + u_{km}(P,Q)\sqrt{\Delta}/2$$

proves (13) and (14). Writing $\alpha^{k+m} = \alpha^k\alpha^m$ as sums of Lucas sequences and comparing the coefficients shows (15) and (16). □

Proposition 3. *Let p be an odd prime, $Q = 1$ and $\gcd(\Delta, p) = 1$. Then the sequence $v_k(P,1) \bmod p$ is periodic and the length of the period divides $p-(\Delta/p)$.*

Proof. α and therefore also α^p are algebraic integers in $\mathbb{Q}(\sqrt{\Delta})$. Thus we have $\alpha^p = (P/2 + \sqrt{\Delta}/2)^p \equiv P/2 + (\sqrt{\Delta})^p/2 \equiv P/2 + \Delta^{(p-1)/2}\sqrt{\Delta}/2 \equiv P/2 + (\Delta/p)\sqrt{\Delta}/2 \pmod p$. Thus if $(\Delta/p) = 1$ then $\alpha^{p-1} \equiv 1 \pmod p$ and if $(\Delta/p) = -1$ then $\alpha^{p+1} \equiv 1 \pmod p$. It follows that the sequence α^k (and therefore also $v_k(P,1)$) is periodic with a period that divides $p - (\Delta/p)$. □

A.2 Elliptic curves

Elliptic curves over $\mathbb{Z}_p$ Let p be a prime greater than 3, and let a and b be two integers such that $4a^3 + 27b^2 \neq 0 \pmod p$. An *elliptic curve $E_p(a,b)$ over the prime field $\mathbb{Z}_p$* is the set of points $(x,y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ satisfying the Weierstraß equation

$$y^2 = x^3 + ax + b \pmod p \tag{17}$$

together with the point at infinity $\mathcal{O}_p$. The points of the elliptic curve $E_p(a,b)$ form an Abelian group under the tangent-and-chord law defined as follows.

(i) $\mathcal{O}_p$ is the identity element, i.e. $\forall \mathbf{P} \in E_p(a,b)$, $\mathbf{P} + \mathcal{O}_p = \mathbf{P}$.
(ii) The inverse of $\mathbf{P} = (x_1, y_1)$ is $-\mathbf{P} = (x_1, -y_1)$.
(iii) Let $\mathbf{P} = (x_1, y_1)$ and $\mathbf{Q} = (x_2, y_2) \in E_p(a,b)$ with $\mathbf{P} \neq -\mathbf{Q}$. Then $\mathbf{P} + \mathbf{Q} = (x_3, y_3)$ where

$$x_3 = \lambda^2 - x_1 - x_2, \tag{18}$$
$$y_3 = \lambda(x_1 - x_3) - y_1, \tag{19}$$

and $$\lambda = \begin{cases} \dfrac{3x_1^2 + a}{2y_1} & \text{if } x_1 = x_2, \\ \dfrac{y_1 - y_2}{x_1 - x_2} & \text{otherwise.} \end{cases}$$

Note that if $\mathbf{P} = (x_1, 0) \in E_p(a,b)$, then $2\mathbf{P} = \mathcal{O}_p$.

Theorem 1 (Hasse). *Let $\#E_p(a,b) = p + 1 - a_p$ denote the number of points in $E_p(a,b)$. Then $|a_p| \leq 2\sqrt{p}$.* □

Complementary group of $E_p(a,b)$ Let $E_p(a,b)$ be an elliptic curve over $\mathbb{Z}_p$. Let D_p be a quadratic non-residue modulo p. The *twist of* $E_p(a,b)$, denoted by $\overline{E_p(a,b)}$, is the elliptic curve given by the (extended) Weierstraß equation

$$D_p y^2 = x^3 + ax + b \tag{20}$$

together with the point at infinity $\mathcal{O}_p$. The sum of two points (that are not inverse of each other) $(x_1,y_1)+(x_2,y_2)=(x_3,y_3)$ can be computed by

$$x_3 = \lambda^2 D_p - x_1 - x_2,$$
$$y_3 = \lambda(x_1 - x_3) - y_1,$$

and $\lambda = \begin{cases} \dfrac{3x_1^2 + a}{2D_p y_1} & \text{if } x_1 = x_2, \\ \dfrac{y_1 - y_2}{x_1 - x_2} & \text{otherwise.} \end{cases}$

Proposition 4. *If* $\#E_p(a,b) = p + 1 - a_p$, *then* $\#\overline{E_p(a,b)} = p + 1 + a_p$.

Proof. Since $\#E_p(a,b) = 1 + \sum_{x\in\mathbb{Z}_p}\big(1 + (x^3 + ax + b/p)\big)$, $a_p = -\sum_{x\in\mathbb{Z}_p}(x^3 + ax + b/p)$. Hence, $\#\overline{E_p(a,b)} = 1 + \sum_{x\in\mathbb{Z}_p}\big(1 - (x^3 + ax + b/p)\big) = 1 + p + a_p$. □

Elliptic curves over $\mathbb{Z}_n$ Let $n = pq$ with p and q two primes greater than 3, and let a and b be two integers such that $\gcd(4a^3 + 27b^2, n) = 1$. An *elliptic curve* $E_n(a,b)$ *over the ring* $\mathbb{Z}_n$ is the set of points $(x,y) \in \mathbb{Z}_n \times \mathbb{Z}_n$ satisfying the Weierstraß equation

$$y^2 = x^3 + ax + b \pmod{n} \tag{21}$$

together with the point at infinity $\mathcal{O}_n$.

Consider the group $\widetilde{E}_n(a,b)$ given by the direct product

$$\widetilde{E}_n(a,b) = E_p(a,b) \times E_q(a,b). \tag{22}$$

By the Chinese remainder theorem there exists a unique point $\mathbf{P} = (x_1, y_1) \in E_n(a,b)$ for every pair of points $\mathbf{P}_p = (x_{1p}, y_{1p}) \in E_p(a,b) \setminus \{\mathcal{O}_p\}$ and $\mathbf{P}_q = (x_{1q}, y_{1q}) \in E_q(a,b) \setminus \{\mathcal{O}_q\}$ such that $x_1 \bmod p = x_{1p}$, $x_1 \bmod q = x_{1q}$, $y_1 \bmod p = y_{1p}$ and $y_1 \bmod q = y_{1q}$. This equivalence will be denoted by $\mathbf{P} = [\mathbf{P}_p, \mathbf{P}_q]$. Since $\mathcal{O}_n = [\mathcal{O}_p, \mathcal{O}_q]$, the group $\widetilde{E}_n(a,b)$ consists of all the points of $E_n(a,b)$ together with a number of points of the form $[\mathbf{P}_p, \mathcal{O}_q]$ or $[\mathcal{O}_p, \mathbf{P}_q]$.

Lemma 1. *The tangent-and-chord addition on* $E_n(a,b)$, *whenever it is defined, coincides with the group operation on* $\widetilde{E}_n(a,b)$.

Proof. Let $\mathbf{P}$ and $\mathbf{Q} \in E_n(a,b)$. Assume $\mathbf{P} + \mathbf{Q}$ is well-defined by the tangent-and-chord rule. Therefore $\mathbf{P} + \mathbf{Q} = [(\mathbf{P}+\mathbf{Q})_p, (\mathbf{P}+\mathbf{Q})_q] = [\mathbf{P}_p + \mathbf{Q}_p, \mathbf{P}_q + \mathbf{Q}_q]$. □

If n is the product of two large primes, it is extremely unlikely that the "addition" is not defined on $E_n(a,b)$. Consequently, computations in $\widetilde{E}_n(a,b)$ can be performed without knowing the two prime factors of n.

On the Importance of Securing Your Bins: The Garbage-Man-in-the-Middle Attack

[Published in T. Matsumoto, Ed., 4th ACM Conference on Computer and Communications Security, pp. 135–141, ACM Press, 1997.]

Marc Joye[1] and Jean-Jacques Quisquater[2]

[1] UCL Crypto Group, Dép. de Mathématique, Université de Louvain
Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium
E-mail: joye@agel.ucl.ac.be
[2] UCL Crypto Group, Lab. de Microélectronique, Université de Louvain
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium
E-mail: jjq@dice.ucl.ac.be

Abstract. In this paper, we address the following problem: " Is it possible to weaken/attack a scheme when a (provably) secure cryptosystem is used? ". The answer is yes. We exploit weak error-handling methods. Our attack relies on the cryptanalyst being able to modify some ciphertext and then getting access to the decryption of this modified ciphertext. Moreover, it applies on many cryptosystems, including RSA, Rabin, LUC, KMOV, Demytko, ElGamal and its analogues, 3-pass system, knapsack scheme, etc...

1 Introduction

At Eurocrypt'96, Coppersmith, Franklin, Patarin and Reiter [4] presented a serious weakness in the basic protocol for encrypting related messages using the RSA public-key cryptosystem [24] with low exponent. Therefore, if an authority uses a RSA-based protocol to distribute secret keys among a group of users, a cryptanalyst can recover the secret keys of the users which have a relatively small public encryption key (typically less than 32 bits).

Suppose that Alice has a small public encryption key and that the authority wants to send her the secret key k_A, then the cryptanalyst does the following. He intercepts the ciphertext corresponding to k_A. So, since Alice does not receive the ciphertext, she asks the authority to re-send it. If the system is not well designed (for example, if the time or an ID number is included before the encryption), then the cryptanalyst applies the attack of Coppersmith *et al.* to recover the secret key from the two corresponding ciphertexts (the intercepted one and its retransmission).

Moreover, even if the authority is more malicious and chooses a clever protocol in order to prevent the previous attack, we will show that a cryptanalyst will be able to recover *all* the secrets keys, and not only the ones encrypted with

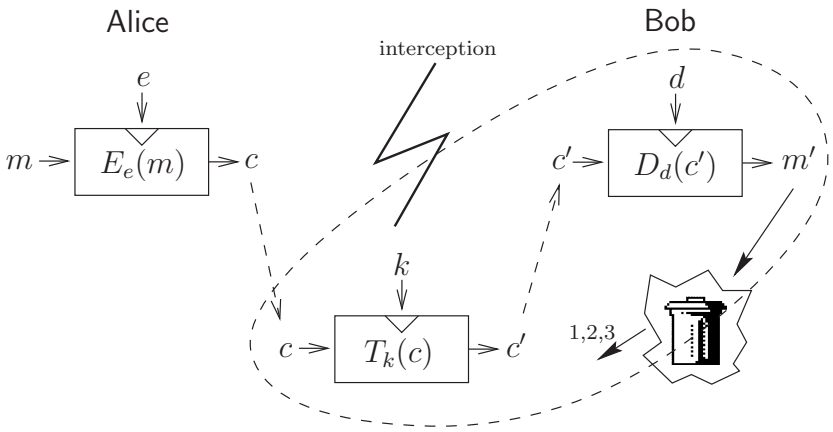


Fig. 1. General description.

a small exponent. Last but not least, our attack is of very general nature. It applies on many public-key cryptosystems, including RSA [24], Rabin [23, 30], LUC [27], KMOV [19], Demytko [8], ElGamal [12] and its analogues [28, 17], 3-pass system [10], knapsack scheme [25], etc...

The basic idea of our attack relies on the possibility to get access to the "bin" of the recipient. In fact, if the cryptanalyst intercepts, transforms and resends a ciphertext, then the corresponding plaintext will be meaningless when the authorized receiver (say, Alice) will decrypt it. So, Alice will discard it. If the cryptanalyst can get access to these discards, he will be able to recover the original plaintext if the transformation is done in a clever way. Such an attack was already be mounted against the RSA by Davida [5]. In many situations, we can get access to the discards, as for example,

- bad implementation of softwares or bad architectures;
- negligent secretaries;
- recovering of a previously deleted message, by a tool like the `<undelete>` command with MS-DOS;

another scenario is to ask the victim to sign the forged messages. Our working hypothesis are thus not unrealistic.

Consequently, the lesson of this paper is that the reader has to be very careful of using a given protocol to distribute secrets keys. In fact, the security of the used cryptosystem is not enough to guarantee the security of the transmitted data; we have also to *really* delete the discards, *i.e.* to secure our "bins".

The paper is organized as follows. In Section 2, we review the attack of Davida and give some (immediate) extensions. Next, we present our attack in Section 3. Section 4 deals with further results. Finally, we conclude in Section 5. The reader who is not familiar with Lucas sequences/elliptic curves over a ring and the resulting cryptosystems may consult the appendix.

2 Attack of Davida

Let p and q be two secret carefully chosen primes, and let $n = pq$ be the corresponding public RSA modulus. The pair (e, d) of public encryption/secret decryption keys of Bob are chosen according to $ed = 1 \pmod{\phi(n)}$.[1]

Suppose Alice wants to send a message m to Bob. Using the Bob's public encryption key, she computes $c = m^e \bmod n$, and sends it to Bob. Then, because only Bob knows the secret decryption key d, he can recover the message $m = c^d \bmod n$.

However, a cryptanalyst (Carol) can also recover the message as follows. She intercepts the ciphertext c, and replaces it by $c' = ck^e \bmod n$ where k is a random number. Then, when Bob will decrypt c', he will compute $m' = {c'}^d \bmod n$. Since the message m' is meaningless, he will discard it. Consequently, if Carol can get access to m', she recovers the original message m by computing

$$m'k^{-1} = {c'}^d k^{-1} = c^d = m \pmod{n}.$$

This attack, first proposed by Davida [5], relies on the homomorphic nature of the RSA. Thus, it can easily be extended to other systems which have the same property [9]. So, DeMillo *et al.* [7] showed that the knapsack system is also vulnerable to this system. The same conclusion holds for the ElGamal cryptosystem and its variants based on Lucas sequences and elliptic curves over a finite field.

[Encryption] For sending a message m to Bob, Alice chooses a random r that is relatively prime to $p - 1$. Then, she looks to the public key of Bob, $y = g^x \bmod p$ where x is the secret key of Bob. She computes the pair $(a = g^r \bmod p, b = my^r \bmod p)$ and sends it to Bob.
[Interception/modification] Carol intercepts the pair (a, b) and replaces it by $(a, b' = bk \bmod p)$ where k is a random. Next, she sends the modified pair to Bob.
[Decryption] Using his secret key x, Bob computes $m' = b'a^{-x} = mk \pmod{p}$. Since m' is meaningless, Bob discards it.
[Recovering] From m', Carol recovers the original message $m = m'k^{-1} \bmod p$.

Fig. 2. Attacking ElGamal.

[1] ϕ denotes the Euler totient function, *i.e.* $\phi(n)$ is the number of positive integers $< n$ and relatively prime to n.

3 Our Attack

3.1 General description

The basic attack of Davida does not apply to all cryptosystems. We need another tool:

Theorem 1 (Lagrange). *Let G be a (multiplicative) finite group of order n. Then each element a of G satisfies to $a^n = 1$.*

Let $E_e(\cdot)$, $D_d(\cdot)$ and $T_k(\cdot)$ be respectively the public encryption function, the corresponding secret decryption function and the cryptanalytic transformation function.

Now, imagine Alice wants to send the message m to Bob. She first computes the ciphertext $c = E_e(m)$, and sends it to Bob. Suppose that the cryptanalyst, Carol, intercepts the ciphertext c, and modifies it into $c' = T_k(c)$ according to the theorem of Lagrange. Next, Bob decrypts c', and gets $m' = D_d(c')$. Since the message m' has no meaning, Bob discards it. Finally, if Carol can get access to m', then she recovers the message m from c, k, c' and m' from a non-trivial relation.

We shall illustrate this attack on two cryptosystems that are not susceptible to the basic attack of Davida: LUC and KMOV/Demytko.[2] However, the same cryptanalysis remains valid for other systems.

3.2 Illustrations

Attacking LUC For the LUC cryptosystem, the enciphering function and the deciphering function are respectively defined by

$$\begin{aligned} E_e : L(D,n) &\to L(D,n), \\ (m,\cdot) &\mapsto (c,\cdot) = (V_e(m,1),\cdot) \bmod n, \end{aligned} \tag{1}$$

$$\begin{aligned} D_d : L(D,n) &\to L(D,n), \\ (c',\cdot) &\mapsto (m',\cdot) = (V_d(c',1),\cdot) \bmod n. \end{aligned} \tag{2}$$

In order to modify the ciphertext c into c', the cryptanalyst uses the transformation function

$$\begin{aligned} T_k : L(D,n) &\to L(D,n), \\ (c,\cdot) &\mapsto (c',\cdot) = (V_k(c,1),\cdot) \bmod n. \end{aligned} \tag{3}$$

The non-trivial relation enabling the attack comes from the following proposition.

[2] See the appendix for a description of these cryptosystems.

Proposition 1. *Let $\{V_k\}$ be the Lucas sequence with parameters P and $Q = 1$. Then*

$$V_k(P,1) = P^k - \sum_{\substack{i=1 \\ i \ odd}}^{k-2} \binom{k}{\frac{k-i}{2}} V_i(P,1). \tag{4}$$

So, it is possible to express recursively $V_k(P,1)$ as a polynomial of degree k in the indeterminate P.

Consequently, to recover the message m, the cryptanalyst, Carol, does the following.

[Lagrange's modification] Carol intercepts $c = V_e(m,1) \bmod n$ and replaces it by $c' = V_k(c,1) \bmod n$, where k is relatively prime to e.

[Recovery of m'] Next, she gets from Bob the value of m', the plaintext corresponding to c':

$$\begin{aligned} m' &= V_d(c',1) = V_{dk}(c,1) = V_{dke}(m,1) \\ &= V_k(m,1) \pmod n. \end{aligned}$$

[Non-trivial relation] She constructs the polynomials $\mathcal{P}, \mathcal{Q} \in \mathbb{Z}_n[x]$ given by

$$\mathcal{P}(x) = V_e(x,1) - c \text{ and } \mathcal{Q}(x) = V_k(x,1) - m',$$

for which m is a root. Then, she computes

$$\mathcal{R} = \gcd(\mathcal{P}, \mathcal{Q}),$$

that is with a very high probability a polynomial of degree 1. So, Carol obtains the value of m by solving $\mathcal{R}$ in x.

Fig. 3. Attacking LUC.

Attacking KMOV/Demytko To present the attack, we need to introduce the division polynomials (see [26], p. 105). They are recursively defined by

$$\begin{aligned} &\Psi_1 = 1, \quad \Psi_2 = 2y, \quad \Psi_3 = 3x^4 + 6ax^2 + 12bx - a^2, \\ &\Psi_4 = 4y(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3), \\ &\Psi_{2m+1} = \Psi_{m+2}\Psi_m^3 - \Psi_{m-1}\Psi_{m+1}^2 \quad (m \geq 2), \\ &2y\,\Psi_{2m} = \Psi_m(\Psi_{m+2}\Psi_{m-1}^2 - \Psi_{m-2}\Psi_{m+1}^2) \quad (m \geq 2). \end{aligned}$$

Proposition 2. *Let $E_n(a,b)$ be an elliptic curve over the ring $\mathbb{Z}_n$. Define the polynomials Φ_k and ω_k by*

$$\begin{aligned} \Phi_k &= x\Psi_k^2 - \Psi_{k+1}\Psi_{k-1}, \\ 4y\,\omega_k &= \Psi_{k+2}\Psi_{k-1}^2 - \Psi_{k-2}\Psi_{k+1}^2. \end{aligned}$$

(a) Ψ_k, Φ_k, $y^{-1}\omega_k$ *(for* k *odd) and* $(2y)^{-1}\Psi_k$, Φ_k, ω_k *(for* k *even) are polynomials in* $\mathbb{Z}[a, b, x, y^2]$. *Hence, by replacing* y^2 *by* x^3+ax+b, *they will be considered as polynomials in* $\mathbb{Z}[a, b, x]$.

(b) As polynomials in x,

$$\Phi_k(x) = x^{k^2} + \textit{lower order terms},$$
$$\Psi_k(x)^2 = k^2 x^{k^2-1} + \textit{lower order terms}$$

are relatively prime polynomials.

(c) If $P \in E_n(a, b)$, *then*

$$[k]P = \left(\frac{\Phi_k(P)}{\Psi_k(P)^2}, \frac{\omega_k(P)}{\Psi_k(P)^3}\right) \pmod{n}.$$

Corollary 1. *There is a univariate polynomial relation between the* x*-coordinates of a point* P *and any of its multiple* kP *over the elliptic curve* $E_n(a, b)$ *given by*

$$\Phi_k\Big(x(P)\Big) - x([k]P)\Psi_k\Big(x(P)\Big)^2 = 0 \pmod{n}.^3 \tag{5}$$

To illustrate the attack, we will only focus on the first coordinate of the points on $E_n(a, b)$. By (5), we have a polynomial relation between the first coordinate of a point and the first coordinate of any multiple of this point.

Let m_x be the message to be encrypted. This message will be represented by the point $M = (m_x, m_y)$ on the elliptic curve $E_n(a, b)$. Then, the cryptanalyst proceeds in a similar way as for LUC to recover the message m_x.

[Lagrange's modification] Carol intercepts $c_x = x([e]M)$ and replaces it by ${c'}_x = x([k]C)$, where k is relatively prime to e.

[Recovery of m'] Next, she gets from Bob the value of ${m'}_x$, the plaintext corresponding to ${c'}_x$:

$${m'}_x = x([d]C') = x([dke]M) = x([k]M).$$

[Non-trivial relation] She constructs the polynomials $\mathcal{P}, \mathcal{Q} \in \mathbb{Z}_n[x]$ given by

$$\mathcal{P}(x) = \Phi_e(x) - c_x\Psi_e(x)^2$$

and

$$\mathcal{Q}(x) = \Phi_k(x) - {m'}_x\Psi_k(x)^2,$$

for which m_x is a root. Then, she computes

$$\mathcal{R} = \gcd(\mathcal{P}, \mathcal{Q}),$$

that is with a very high probability a polynomial of degree 1. So, Carol obtains the value of m_x by solving $\mathcal{R}$ in x.

Fig. 4. Attacking KMOV/Demytko.

[3] $x(P)$ denotes the x-coordinate of the point P.

3.3 Analysis

In [22], Patarin estimates that the computation of a GCD may be done as long as the exponent of the polynomial is less than 32-bit long. So, unlike the basic attack of Davida against the RSA, from relation (4), our attack applies on LUC only if the public encryption exponent e has length less than 32 bits. Moreover, when Alice encrypts a message with the KMOV or Demytko cryptosystem with a public exponent e, the polynomial relation (5) is of order e^2 instead of e as in LUC. Therefore, our attack is useless against KMOV/Demytko if the encryption exponent e has length greater than 16 bits.

To overcome this drawback, the cryptanalyst (Carol) has to apply the attack in two times. We shall illustrate the technique on KMOV/Demytko. The notation is the same as used in paragraph 3.2.

Let $M = (m_x, m_y)$ and $C = [e]M = (c_x, c_y)$ be two points on the elliptic curve $E_n(a, b)$, where e is the public encryption key of Bob. Let m_x be the message to be encrypted. Then, the attack goes as follows. Carol intercepts c_x, and replaces it by $c'_{1,x} = x([k_1]C)$. Next, Bob computes

$$m'_{1,x} = x([d]C'_1) = x([dk_1e]M) = x([k_1]M). \tag{6}$$

Carol chooses k_2 (relatively prime to k_1), and sends $c'_{2,x} = x([k_2]C)$ to Bob. Then, Bob computes

$$m'_{2,x} = x([d]C'_2) = x([dk_2e]M) = x([k_2]M). \tag{7}$$

Therefore, from relations (6) and (7), Carol forms the polynomials $\mathcal{P}$ and $\mathcal{Q} \in \mathbb{Z}_n[x]$ given by

$$\mathcal{P}(x) = \Phi_{k_1}(x) - m'_{1,x}\Psi_{k_1}(x)^2$$

and

$$\mathcal{Q}(x) = \Phi_{k_2}(x) - m'_{2,x}\Psi_{k_2}(x)^2,$$

for which m_x is a root. So, if k_1 and k_2 are "small" (typically less than 16-bit long), then by solving the polynomial $\mathcal{R} = \gcd(\mathcal{P}, \mathcal{Q})$, Carol obtains the message m_x.

4 Further Results

4.1 Substituting the authority

Imagine we deal with a key distribution scheme. If the cryptanalyst intercepts the encrypted key c sent by the authority to Bob, and modifies it into c', then Bob will discover that the key is corrupted when he will decrypt it. Therefore, he will ask to the authority to re-send the key. If now, the cryptanalyst plays the role of the authority, *i.e.* if he sends the encrypted key c, then the authority will never know that a pirate knows the secret key of Bob. The cryptanalyst behaves thus transparently for the authority.

4.2 Concealing the cryptanalyst

The aim of the cryptanalyst is to modify the ciphertext c in such a way that Bob is not able to make the difference between his modification and noise (error of transmission) on the public channel. Consequently, he has to modify c in an apparently random way. So, he has to use "large" exponent for the transformation or to use the basic attack of Davida (when applicable).

We shall illustrate this topic on LUC. We use the same notation as in 3.2. Imagine that the cryptanalyst, Carol, chooses a small exponent k in order to speed up the computation of the GCD. Then, Bob can "prove" that somebody (*i.e.* Carol) modified the ciphertext c into c' by recovering k. He has just to compare (modulo n) $V_j(m,1)$ with m', for $j = 2, 3, \ldots, k$. To prevent this, Carol has to choose a relatively large exponent k. However, in order to recover the plaintext, she has to get access two or three times to the bin.

Let m be the message that Alice wants to send to Bob, and let $c = V_e(m,1) \mod n$ be the corresponding ciphertext, where e is the public key of Bob. Then, the attack is the following.

Carol intercepts c, and replaces it by $c'_1 = V_{k_1}(c,1) \bmod n$. Bob receives c'_1, and decrypts it with its secret key d as $m'_1 = V_d(c'_1,1) \bmod n$. Bob asks Alice to re-send c, and Carol sends $c'_2 = V_{k_2}(c,1) \bmod n$. When Bob computes $m'_2 = V_d(c'_2,1) \bmod n$, he finds a meaningless message. So, he asks to Alice to send a third time the ciphertext c. Next, Carol sends $c'_3 = V_{k_3}(c,1) \bmod n$. Bob decrypts c'_3 to $m'_3 = V_d(c'_3,1) \bmod n$, ...

Now, from the discards m'_i $(i = 1, 2(,3))$, Carol can recover the original message m if k_1, k_2 and k_3 and correctly chosen. This can for instance be done by selecting

$$k_1 = rst, k_2 = ru \text{ and } k_3 = sv,$$

where r and s are small, and $\gcd(st,u) = \gcd(rt,v) = 1$. Note that t, u and v must be sufficiently large to disable Bob to distinguish noise with piracy on the public channel.

From our choices on the transformation keys k_i, Carol obtains

$$\begin{aligned} m'_1 = V_{rst}(m,1) &= V_{st}\Big(V_r(m,1),1\Big) \\ &= V_{rt}\Big(V_s(m,1),1\Big) \pmod n, \\ m'_2 = V_{ru}(m,1) &= V_u\Big(V_r(m,1),1\Big) \pmod n, \\ m'_3 = V_{sv}(m,1) &= V_v\Big(V_s(m,1),1\Big) \pmod n. \end{aligned}$$

Next, she forms the polynomials $\mathcal{Q}_{1_r}, \mathcal{Q}_2 \in \mathbb{Z}_n[y]$ given by

$$\mathcal{Q}_{1_r}(y) = V_{st}(y,1) - m'_1 \quad \text{and} \quad \mathcal{Q}_2(y) = V_u(y,1) - m'_2,$$

for which $V_r(m,1)$ is a root.
Hence, by computing $\mathcal{R} = \gcd(\mathcal{Q}_{1_r}, \mathcal{Q}_2)$, she gets (with a high probability) a polynomial of degree 1, for which $m_r = V_r(m,1) \bmod n$ is the root.

Now, if e is small, Carol recovers the original message m by constructing $\mathcal{P}, \mathcal{R} \in \mathbb{Z}_n[x]$ given by

$$\mathcal{P}(x) = V_e(x, 1) - c \quad \text{and} \quad \mathcal{R}(x) = V_r(x, 1) - m_r,$$

and by computing $\gcd(\mathcal{P}, \mathcal{Q})$, she recovers the original message m as explained before.

Otherwise, she constructs the polynomials $\mathcal{Q}_{1_s}, \mathcal{Q}_3 \in \mathbb{Z}_n[z]$ as

$$\mathcal{Q}_{1_s}(z) = V_{rt}(z, 1) - m'_1 \quad \text{and} \quad \mathcal{Q}_3(z) = V_v(z, 1) - m'_3,$$

for which $V_s(m, 1)$ is a root.
Hence, by computing $\mathcal{S} = \gcd(\mathcal{Q}_{1_s}, \mathcal{Q}_3)$, she gets (with a high probability) a polynomial of degree 1, for which $m_s = V_s(m, 1) \bmod n$ is the root. Next, from polynomials $\mathcal{R}, \mathcal{S} \in \mathbb{Z}_n[x]$ given by $\mathcal{R}(x) = V_r(x, 1) - m_r$ and $\mathcal{S}(x) = V_s(x, 1) - m_s$, she computes $\gcd(\mathcal{Q}, \mathcal{R})$ and recovers the message m.

Remark 1. It is possible to speed up the computation by using the ideas developed in [1] (see [16] for KMOV/Demytko). This will be done in a future work.

4.3 Combinations/Broadcast encryption

There are basically three ways for a cryptanalyst to recover a message

1. to force the retransmission;
2. to have a look in the bin;
3. to ask a signature.

Therefore, by combining these methods, it is possible to recover the message.

Furthermore, if the same message is broadcasted to several people, the security is compromised if only two or three persons are negligent (*i.e.* do not protect their bins).

4.4 Rabin-type cryptosystems

The Rabin-type cryptosystems [23, 30] have the advantage to be provably as intractable as factorization. However, since they are completely insecure against a chosen-ciphertext attack, our attack enables to recover the factorization of the public modulus n.

5 Conclusions

In this paper, we have shown that if one can get access to the "bin", then the security is compromised. We have illustrated our purpose with the Lagrange theorem. This is perhaps not the best tool. The basic aim of the paper was to warn the user that

> *"Even if a secure cryptosystem is used for a given application, this doesn't mean a pirate cannot mount a successful attack."*

This was shown to quite a lot of cryptosystems.

Acknowledgments We are grateful to George Davida for sending a copy of his attack [5]. Thanks also to Moti Yung for suggesting the title.

References

1. Bleichenbacher, D., Bosma, W., and Lenstra, A. K. Some remarks on Lucas-based cryptosystems. In *Advance in Cryptology – CRYPTO '95* (1995), D. Coppersmith, Ed., vol. 963 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 386–396.
2. Bressoud, D. M. *Factorization and primality testing.* Undergraduate Texts in Mathematics. Springer-Verlag, 1989.
3. Cohen, H. *A course in computational algebraic number theory*, vol. 138 of *Graduate Texts in Mathematics.* Springer-Verlag, 1993.
4. Coppersmith, D., Franklin, M., Patarin, J., and Reiter, M. Low exponent RSA with related messages. In *Advance in Cryptology – EUROCRYPT '96* (1996), U. Maurer, Ed., vol. 1070 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–9.
5. Davida, G. Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem. Tech. Rep. TR-CS-82-2, Dept. of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, USA, Oct. 1982.
6. DeLaurentis, J. M. A further weakness in the common modulus protocol for the RSA cryptoalgorithm. *Cryptologia 8*, 3 (July 1984), 253–259.
7. DeMillo, R., Lynch, N. A., and Merritt, M. J. Cryptographic protocols. In *Proc. SIGACT Conf.* (1982).
8. Demytko, N. A new elliptic curve based analogue of RSA. In *Advance in Cryptology – EUROCRYPT '93* (1993), T. Helleseth, Ed., vol. 765 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 40–49.
9. Denning, D. E. Digital signatures with RSA and other public-key cryptosystems. *Communications of the ACM 27*, 4 (Apr. 1984), 388–392.
10. Desmedt, Y., and Odlyzko, A. M. A chosen text attack on the RSA cryptosystem and some discrete logarithms schemes. In *Advance in Cryptology – CRYPTO '85* (1986), H. C. Williams, Ed., vol. 218 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 516–521.
11. Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory IT-26*, 6 (Nov. 1976), 644–654.
12. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory IT-31*, 4 (July 1985), 469–472.
13. Håstad, J. On using RSA with low exponent in a public key network. In *Advance in Cryptology – CRYPTO '85* (1986), H. C. Williams, Ed., vol. 218 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 404–408.
14. Husemöller, D. *Elliptic curves*, vol. 111 of *Graduate Texts in Mathematics.* Springer-Verlag, 1987.
15. Joye, M., and Quisquater, J.-J. Protocol failures for RSA-like functions using Lucas sequences and elliptic curves over a ring. Pre-proceedings of the 1996 Cambridge Workshop on Security Protocols, Apr. 1996.
16. Kaliski Jr, B. S. A chosen message attack on Demytko's elliptic curve cryptosystem. To appear in Journal of Cryptology.

17. Koblitz, N. Elliptic curve cryptosystems. *Mathematics of Computation 48* (1987), 203–209.
18. Koblitz, N. *A course in number theory and cryptography*, 2nd ed., vol. 114 of *Graduate Texts in Mathematics*. Springer-Verlag, 1994.
19. Koyama, K., Maurer, U. M., Okamoto, T., and Vanstone, S. A. New public-key schemes based on elliptic curves over the ring $\mathbb{Z}_n$. In *Advance in Cryptology – CRYPTO '91* (1991), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 252–266.
20. Menezes, A. J. *Elliptic curve public key cryptosystems*. Kluwer Academic Publishers, 1993.
21. Miller, V. S. Use of elliptic curves in cryptography. In *Advance in Cryptology – CRYPTO '85* (1986), H. C. Williams, Ed., vol. 218 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 417–426.
22. Patarin, J. Some serious protocol failures for RSA with exponent e of less than $\simeq 32$ bits. Presented at the conference of cryptography, CIRM Luminy, France, Sept. 1995.
23. Rabin, M. O. Digital signatures and public-key functions as intractable as factorization. Tech. Rep. MIT/LCS/TR-212, MIT Laboratory for Computer Science, Jan. 1979.
24. Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (Feb. 1978), 120–126.
25. Shamir, A. A fast signature scheme. Tech. Rep. MIT/LCS/TM-107, MIT Lab. for Computer Science, Cambridge, Mass., July 1978.
26. Silverman, J. *The arithmetic of elliptic curves*, vol. 106 of *Graduate Texts in Mathematics*. Springer-Verlag, 1986.
27. Smith, P. LUC public-key encryption. *Dr. Dobb's Journal* (Jan. 1993), 44–49.
28. Smith, P., and Skinner, C. A public-key cryptosystem and a digital signature based on the Lucas function analogue to discrete logarithms. In *Advance in Cryptology – ASIACRYPT '94* (1995), J. Pieprzyk, Ed., vol. 917 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 357–364.
29. Smith, P. J., and Lennon, M. J. J. LUC: A new public key system. In *Ninth IFIP Symposium on Computer Security* (1993), E. G. Douglas, Ed., Elsevier Science Publishers, pp. 103–117.
30. Williams, H. C. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory IT-26*, 6 (Nov. 1980), 726–729.

A Lucas Sequences

In 1993, Smith [27] proposed to use the Lucas sequences in order to construct a public-key cryptosystem.

The reader who is not familiar with Lucas sequences may found an elementary introduction in [2].

A.1 Definition

Let D be an integer congruent to 0 or 1 modulo 4, which is a non-square, and let P be an integer with the same parity as D such that 4 divides $P^2 - D$. The

Lucas sequences $\{U_i\}_{i\geq 0}$ *and* $\{V_i\}_{i\geq 0}$ are defined by

$$V_i + U_i\sqrt{D} = 2^{1-i}(P + \sqrt{D})^i. \tag{8}$$

It can easily be shown that the numbers U_i and V_i satisfy the following relations:

$$\begin{gathered} U_0 = 0, U_1 = 1, V_0 = 2, V_1 = P, \\ U_{i+j} = U_i V_j - Q^j U_{i-j}, \\ V_{i+j} = V_i V_j - Q^j V_{i-j}, \end{gathered}$$

where $Q = (P^2 - D)/4$.

Moreover, if we take $Q = 1$ in the definition of the Lucas sequences and if n is relatively prime to $2D$, then we call $L(D, n)$ the group of Lucas sequences (see [2], p. 196). The elements of $L(D, n)$ are the pairs (V_i, U_i) modulo n obtained from the Lucas sequences $\{V_i\}$ and $\{U_i\}$ with parameter $Q = 1$. The identity element is $(2, 0)$. Two elements $(V_k, U_k), (V_m, U_m) \in L(D, n)$ are composed according to the law ∂ defined by $(V_k, U_k)\partial(V_m, U_m) = (V_{k+m}, U_{k+m})$.

Theorem 2. *If* $p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ *is the prime factorization of* n*, then the order of* $L(D, n)$ *is given by*[4]

$$\begin{aligned} \#L(D, n) = (p_1 - \left(\frac{D}{p_1}\right))p_1^{e_1-1}(p_2 - \left(\frac{D}{p_2}\right))p_2^{e_2-1} \cdots \\ (p_r - \left(\frac{D}{p_r}\right))p_r^{e_r-1}. \end{aligned}$$

Corollary 2. *Let* $\Psi(p) = p - (D/p)$*, where* p *is a prime which does not divide* $2D$*. Then, for any integer* k*,*

$$V_{k\Psi(p)+1}(P, 1) \equiv P \pmod{p}.$$

A.2 LUC

The LUC cryptosystem is based on the following proposition.

Proposition 3. *Let* $V_i(P, 1)$ *be the* i^{th} *terms of the Lucas sequence* $\{V_i\}$ *with parameters* P *and* $Q = 1$*. Then,*

$$V_{mk}(P, 1) = V_m\Big(V_k(P, 1), 1\Big).$$

[4] (D/p) denotes the Legendre symbol and is equal to 1 or -1 if D is a quadratic residue modulo p or not.

Each user chooses two secret large primes p and q, and publishes their product $n = pq$. Next, he chooses a public encryption key e which is relatively prime to $(p-1)$, $(p+1)$, $(q-1)$ and $(q+1)$.

To send an encrypted message m to Bob, Alice looks to Bob's public key e, and using the Lucas sequence $\{V_i\}$, she forms the ciphertext $c = V_e(m,1) \bmod n$. Since Bob knows the factors p and q, he is able to compute the secret decryption key d according to $ed = 1 \bmod \Psi(n)$, where $\Psi(n) = \mathrm{lcm}(p-(D/p), q-(D/q))$. Therefore, he recovers the message m by computing

$$V_d\Big(V_e(m,1),1\Big) = V_{ed}(m,1) = V_1(m,1) = m \pmod n.$$

B Elliptic Curves

In 1991, Koyama, Maurer, Okamoto and Vanstone [19] proposed new trapdoor one-way functions (TOFs) based on elliptic curves over the ring $\mathbb{Z}_n$. We call KMOV the resulting cryptosystem. Two years later, Demytko [8] proposed another cryptosystem based on the same structure.

For an introduction to elliptic curves, the reader may consult [20, 18]. A deeper study may be found in [26].

B.1 Elliptic curves over $\mathbb{Z}_n$

Let $\mathbb{F}_p$ be a prime field of characteristic $p \neq 2, 3$, and let $a, b \in \mathbb{F}_p$ such that $4a^2 + 27b^3 \neq 0$. An elliptic curve over $\mathbb{F}_p$ with parameters a and b is the set of points (x, y) satisfying the Weierstraß equation

$$y^2 = x^3 + ax + b, \tag{9}$$

together with a special point $\mathcal{O}$ called the point at infinity. Such a curve will be denoted by $E_p(a, b)$.

Let $P, Q \in E_p(a, b)$, ℓ be the line connecting P and Q (tangent line if $P = Q$), and T be the third point of intersection of ℓ with $E_p(a, b)$. Let ℓ' be the line connecting T and $\mathcal{O}$. Then $P + Q$ is the point such that ℓ' intersects $E_p(a, b)$ at $T, \mathcal{O}$ and $P + Q$. This composition law makes $E_p(a, b)$ into an Abelian group with identity element $\mathcal{O}$.

An elliptic curve over the ring $\mathbb{Z}_n$ is defined in the same way, except that all computations are done modulo n instead of modulo p. However the resulting structure is not a group, but a proposition similar to the Lagrange theorem holds.

Proposition 4. *Let $n = pq$, where p and q are prime. Let $E_n(a, b)$ be an elliptic curve such that $\gcd(4a^3 + 27b^2, n) = 1$, and let*

$$N_n = \mathrm{lcm}(\#E_p(a, b), \#E_q(a, b)). \tag{10}$$

Then, for any $P \in E_n(a, b)$, and any integer k,

$$[kN_n + 1]P = P$$

on $E_n(a,b)$.[5]

Remark 2. As mentioned in [19], it is also possible to define an elliptic curve over a ring such that the resulting structure is a group.

B.2 KMOV/Demytko

The KMOV and the Demytko cryptosystems are both based on elliptic curves over the ring $\mathbb{Z}_n$, where n is the product of two primes p and q.

Imagine Alice wants to send a message m to Bob. She first represents the message m by a publicly known way as a point $M = (m_x, m_y)$ of the elliptic curve $E_n(a,b)$. Then, by using the public encryption key of Bob, she computes $C = (c_x, c_y) = [e]M$ over $E_n(a,b)$.

To recover the message m, Bob computes $M = [d]C$ over $E_n(a,b)$ with his secret decryption d according to $ed = 1 \pmod{N_n}$.

In the KMOV cryptosystem, the primes p and q are both congruent to 2 modulo 3, and the parameter a is equal to 0. In that case, we can show that

$$N_n = \operatorname{lcm}(\#E_p(0,b), \#E_q(0,b)) = \operatorname{lcm}(p+1, q+1).$$

Since N_n does not depend on b, the parameter b is chosen according to

$$b = m_y^2 - m_x^3 \bmod n.$$

With the Demytko cryptosystem, in order to decrypt $C = (c_x, c_y)$, Bob computes $[d_i]C$, where the decryption key $d = d_i$ is chosen so that

$$ed_i \equiv 1 \pmod{N_{n,i}} \quad (i = 1, \ldots, 4),$$

$$\text{with } \begin{cases} N_{n,1} = \operatorname{lcm}(\#E_p(a,b), \#E_q(a,b)) \\ \qquad \text{if } (w/p) = 1 \text{ and } (w/q) = 1 \\ N_{n,2} = \operatorname{lcm}(\#E_p(a,b), \#\overline{E_q(a,b)}) \\ \qquad \text{if } (w/p) = 1 \text{ and } (w/q) \neq 1 \\ N_{n,3} = \operatorname{lcm}(\#\overline{E_p(a,b)}, \#E_q(a,b)) \\ \qquad \text{if } (w/p) \neq 1 \text{ and } (w/q) = 1 \\ N_{n,4} = \operatorname{lcm}(\#\overline{E_p(a,b)}, \#\overline{E_q(a,b)}) \\ \qquad \text{if } (w/p) \neq 1 \text{ and } (w/q) \neq 1 \end{cases},$$

and $w = c_x^3 + ac_x + b \bmod n$.[6]

Remark 3. In the KMOV cryptosystem, it is also possible to work on the elliptic curve $E_n(a,0)$, if one chooses p and q both congruent to 3 modulo 4.

Remark 4. The computation of the second coordinate can be avoided if the algorithm described in [2, pp. 211-216] is used.

[5] $[k]P$ means $\underbrace{P + P + \cdots + P}_{k \text{ times}}$.

[6] $\overline{E_p(a,b)}$ denotes the complementary group of $E_p(a,b)$, *i.e.* the set of points satisfying Weierstraß equation (9) together with $\mathcal{O}$, where y is of the form $u\sqrt{v}$ with v is a fixed quadratic non-residue modulo p and $u \in \mathbb{F}_p$.

Reducing the Elliptic Curve Cryptosystem of Meyer-Müller to the Cryptosystem of Rabin-Williams

[Published in *Designs, Codes and Cryptography* **14**(1):53–56, 1998.]

Marc Joye[1] and Jean-Jacques Quisquater[2]

[1] Dép. de Mathématique (AGEL), Université catholique de Louvain
Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium
joye@agel.ucl.ac.be
[2] Dép. d'Électricité (DICE), Université catholique de Louvain
Place de Levant 3, B-1348 Louvain-la-Neuve, Belgium
jjq@dice.ucl.ac.be

Abstract. At Eurocrypt '96, Meyer and Müller presented a new Rabin-type cryptosystem based on elliptic curves. In this paper, we will show that this cryptosystem may be reduced to the cryptosystem of Rabin-Williams.

Keywords. Cryptography, elliptic curves, Rabin-type cryptosystems.

1 Introduction

In 1991, Koyama, Maurer, Okamoto and Vanstone [5] pointed out the existence of new one-way trapdoor functions similar to the RSA [10] on elliptic curves over a ring. At Eurocrypt '96, Meyer and Müller [7] presented another elliptic RSA-type cryptosystem with a public encryption exponent equal to 2. We will show that this cryptosystem may be reduced to the cryptosystem of Rabin-Williams [9, 11]. This has a lot of consequences. For example, Meyer and Müller claimed that 11 messages are required to mount successfully the so-called low exponent attack against their cryptosystem. However, since the system is reducible to the Rabin-Williams' one, only two messages [1] are required by using the algorithm of Coppersmith [2].

The remainder of the paper is organized as follows. Section 2 describes the cryptosystem of Meyer and Müller. In Section 3, we show how it may be reduced to the cryptosystem of Rabin-Williams. Finally, we conclude in Section 4.

2 Elliptic Curve Cryptosystem of Meyer-Müller

In this section, we describe succinctly the cryptosystem of Meyer and Müller. For a detailed description, we refer to the original paper (see [7]).

Let n be the product of two large secret primes p and q, both congruent to 11 modulo 12. Consider the elliptic curve E over the ring $\mathbb{Z}/n\mathbb{Z}$ given by the Weierstraß equation:

$$E : y^2 = x^3 + ax + b.$$

2.1 Encryption procedure

Assume Alice wants to send the message m to Bob. First, she randomly chooses $\lambda \in \mathbb{Z}/n\mathbb{Z} - \{0\}$, and sets $P = (m^2, \lambda m^3) \in E$. Next, she sets $a = \lambda^3$, and computes $b = (\lambda^2 - 1)m^6 - am^2$. Finally, she computes the point $Q = 2P$ on the curve E, and sends the corresponding ciphertext consisting of

$$a, b, x(Q), t = \left(\frac{y(Q)}{n}\right), l = \mathrm{lsb}\left(y(Q)\right).$$

2.2 Decryption procedure

Since Bob knows the factorization of n, he can recover the message m as follows. He first computes the unique square root $y(Q)$ of $x(Q)^3 + ax(Q) + b$, with type t and lsb l. Next, he computes the set

$$I = \left\{1 \leq i \leq s \mid 2P_i = Q \text{ and } a^2 = y(P_i)^6 x(P_i)^{-9}\right\}.$$

If $\#I = 1$, then the message is given by $m = y(P_1)^3 x(P_1)^{-4} a^{-1}$.

3 Analysis

We can easily determine two polynomials $\mathcal{P}_1$ and $\mathcal{P}_2 \in \mathbb{Z}/n\mathbb{Z}[X]$ for which m^2 is a root.

Since the point $P = (m^2, \lambda m^3)$ is on the curve, we have

$$\lambda^2 m^6 = m^6 + am^2 + b. \tag{1}$$

So, by cubing (1), and by replacing m^2 by X, we obtain the first polynomial

$$\begin{aligned}\mathcal{P}_1(X) &= \lambda^6 X^9 - (X^3 + aX + b)^3 \\ &= (a^2 - 1)X^9 - 3aX^7 - 3bX^6 - 3a^2X^5 - 6abX^4 - (a^3 + 3b^2)X^3 \\ &\quad -3a^2bX^2 - 3ab^2X - b^3. \end{aligned} \tag{2}$$

The second polynomial is constructed from the first coordinate of the point $Q = 2(m^2, \lambda m^3)$, which is given by

$$x(Q) = \frac{(3m^4 + a)^2}{4\lambda^2 m^6} - 2m^2.$$

Hence, with equation (1), we have

$$\begin{aligned}\mathcal{P}_2(X) &= \big(x(Q) + 2X\big)(4\lambda^2 X^3) - (3X^2 + a)^2 \\ &= \big(x(Q) + 2X\big)4(X^3 + aX + b) - (3X^2 + a)^2 \\ &= -X^4 + 4x(Q)X^3 + 2aX^2 + \big(8b + 4a\,x(Q)\big)X - a^2 + 4b\,x(Q). \quad (3)\end{aligned}$$

Since m^2 is a root of $\mathcal{P}_1$ and $\mathcal{P}_2$, m^2 will be a root of

$$\mathcal{R} = \gcd(\mathcal{P}_1, \mathcal{P}_2), \quad (4)$$

which is, with a very high probability [8, 3], a polynomial of degree 1. Thus, by solving this polynomial in X, we obtain the value of m^2.

4 Conclusion

We showed that the system of Meyer and Müller may be reduced to the cryptosystem of Rabin-Williams, because it enables to recover the value of m^2 from the Meyer-Müller's cryptogram corresponding to the message m.

Acknowledgements

We are grateful to Bernd Meyer and Volker Müller for providing us information about their cryptosystem. We are also grateful to Kenji Koyama for pointing out the existence of another Rabin-type cryptosystem based on elliptic curves [6]. We finally thank Richard Pinch for his careful reading of a preliminary version of this paper.

A Index to Notations

Formal symbolism	Meaning
E	elliptic curve
$x(P)$	x-coordinate of point $P \in E$
$y(P)$	y-coordinate of point $P \in E$
$\mathrm{lsb}(a)$	least significant bit of a
(a/n)	Jacobi's symbol of a modulo n
$\mathbb{Z}_n$	ring of integers modulo n

References

1. D. Bleichenbacher. Personal communication.
2. D. Coppersmith. Finding a small root of a univariate modular equation. In U. Maurer, editor, *Advances of Cryptology – Eurocrypt'96*, volume 1070 of *Lectures Notes in Computer Science*, pages 155–165. Springer-Verlag, 1996.

3. D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter. Low-exponent RSA with related messages. In U. Maurer, editor, *Advances of Cryptology – Eurocrypt'96*, volume 1070 of *Lectures Notes in Computer Science*, pages 1–9. Springer-Verlag, 1996.
4. M. Joye and J.-J. Quisquater. On the cryptosystem of Chua and Ling. Technical Report CG-1997/4, UCL Crypto Group, April 1997.
5. K. Koyama, U. Maurer, T. Okamoto, and S. Vanstone. New public-key schemes based on elliptic curves over the ring $\mathbb{Z}_n$. In J. Feigenbaum, editor, *Advances of Cryptology – Crypto'91*, volume 576 of *Lectures Notes in Computer Science*, pages 252–256. Springer-Verlag, 1991.
6. H. Kuwakado and K. Koyama. A uniquely decipherable Rabin-type scheme over elliptic curves. Technical Report ISEC95-33 (in Japanese), IEICE, December 1995.
7. B. Meyer and V. Müller. A public key cryptosystem based on elliptic curves over $\mathbb{Z}/n\mathbb{Z}$ equivalent to factoring. In U. Maurer, editor, *Advances of Cryptology – Eurocrypt'96*, volume 1070 of *Lectures Notes in Computer Science*, pages 49–59. Springer-Verlag, 1996.
8. J. Patarin. Some serious protocol failures for RSA with exponent e of less than $\simeq$ 32 bits. In *Proceedings of the Conference of Cryptography*, Luminy, France, September 1995.
9. M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT, Laboratory for Computer Science, January 1979.
10. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
11. H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, IT-26(6):726–729, November 1980.

New Attacks on PKCS #1 v1.5 Encryption

[Published in B. Preneel, Ed., *Advances in Cryptology – EUROCRYPT 2000*, vol. 1807 of *Lecture Notes in Computer Science*, pp. 369–381, Springer-Verlag, 2000.]

Jean-Sébastien Coron[1,3], Marc Joye[2], David Naccache[3], and Pascal Paillier[3]

[1] École Normale Supérieure
45 rue d'Ulm, 75005 Paris, France
coron@clipper.ens.fr
[2] Gemplus Card International
Parc d'Activités de Gémenos, B.P.100, 13881 Gémenos, France
marc.joye@gemplus.com
[3] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{jean-sebastien.coron, david.naccache, pascal.paillier}@gemplus.com

Abstract. This paper introduces two new attacks on PKCS#1 v1.5, an RSA-based encryption standard proposed by RSA Laboratories. As opposed to Bleichenbacher's attack, our attacks are chosen-plaintext only, i.e. they do *not* make use of a decryption oracle. The first attack applies to small public exponents and shows that a plaintext ending by sufficiently many zeroes can be recovered efficiently when two or more ciphertexts corresponding to the same plaintext are available. We believe the technique we employ to be of independent interest, as it extends Coppersmith's low-exponent attack to certain length parameters. Our second attack is applicable to *arbitrary* public exponents, provided that most message bits are zeroes. It seems to constitute the first chosen-plaintext attack on an RSA-based encryption standard that yields to practical results for any public exponent.

Keywords. RSA, PKCS #1 v1.5 encryption, chosen plaintext attack.

1 Introduction

PKCS stands for *Public-Key Cryptography Standards*. It is a large corpus of specifications covering RSA encryption [13], Diffie-Hellman key agreement, password-based encryption, syntax (extended-certificates, cryptographic messages, private-key information and certification requests) and selected attributes. Historically, PKCS was developed by RSA Laboratories, Apple, Digital, Lotus, Microsoft, MIT, Northern Telecom, Novell and Sun. The standards have been regularly updated since. Today, PKCS has become a part of several standards and of a wide range of security products including Internet Privacy-Enhanced Mail.

Amongst the PKCS collection, PKCS#1 V1.5 describes a particular encoding method for RSA encryption called `rsaEncryption`. In essence, the enveloped data is first encrypted under a randomly chosen key K using a symmetric block-cipher (e.g. a triple DES in CBC mode) then K is RSA-encrypted with the recipient's public key.

In 1998, Bleichenbacher [2] published an adaptive chosen-ciphertext attack against PKCS#1 V1.5 capable of recovering arbitrary plaintexts from a few hundreds of thousands of ciphertexts. Although active adversary models are generally viewed as theoretical issues,[1] Bleichenbacher's attack makes use of an oracle that only detects conformance with respect to the padding format, a real-life assumption leading to a practical threat. PKCS#1 was subsequently updated in the release 2.0 [15] and patches were issued to users wishing to continue using the old version of the standard.

Independently, there exist several well-known chosen-plaintext attacks on RSA-based encryption schemes [8, 5]. These typically enable an attacker to decrypt ciphertexts at moderate cost without requiring to factor the public modulus. The most powerful cryptanalytic tool applicable to low exponent RSA is probably the one based on a theorem due to Coppersmith [6]. As a matter of fact, one major purpose of imposing a partially random padding form to messages, besides attempting to achieve a proper security level such as indistinguishability, is to render the whole encryption scheme resistant against such attacks.

This paper shows that, despite these efforts, chosen-plaintext attacks are actually sufficient to break PKCS#1 V1.5 even in cases when Coppersmith's attack does not apply. We introduce new cryptanalytic techniques allowing an attacker to retrieve plaintexts belonging to a certain category, namely messages ending by a required minimum number of zeroes. The first attack requires two or more ciphertexts corresponding to the same plaintext. Although specific, our attacks only require a *very* small amount of ciphertexts (say ten of them), are completely independent from the public modulus given its size and, moreover, are fully practical for usual modulus sizes.

The rest of this paper is divided as follows. Section 2 introduces a new low-exponent attack for which we provide a comparison with Coppersmith's attack in Section 3. Section 4 shows how to deal with arbitrary public exponents while staying within the chosen-plaintext attack model. Counter-measures are discussed in Section 5. For completeness, Appendix reports practical experiments of our technique performed on 1024-bit ciphertexts.

2 Our Low-Exponent Chosen-Plaintext Attack

We briefly recall the PKCS#1 V1.5 encoding procedure [14]. Let $\{n, e\}$ be an RSA public key and d be the corresponding secret key. Denoting by k the byte-length of n, we have $2^{8(k-1)} \leq n < 2^{8k}$. A message m of size $|m|$ bytes with $|m| \leq k-11$

[1] Chosen-ciphertext attacks require the strong assumption that the adversary has a complete access to a decryption oracle.

is encrypted as follows. A padding r' consisting of $k-3-|m| \geq 8$ nonzero bytes is generated at random. Then the message m gets transformed into:

$$\text{PKCS}(m, r') = \mathtt{0002}_{16} \| r' \| \mathtt{00}_{16} \| m \,,$$

and encrypted to form the ciphertext:

$$c = \text{PKCS}(m, r')^e \bmod n \ .$$

Letting $r = (\mathtt{0002}_{16} \| r')$, we can write $\text{PKCS}(m, r') = r\, 2^\beta + m$ with $\beta = 8|m| + 8$. Now assume that m has its least Z significant bits equal to zero. Hence, we can write $m = \bar{m}\, 2^Z$ and subsequently:

$$\text{PKCS}(m, r') = 2^Z (r\, 2^{\beta - Z} + \bar{m}) \ .$$

From two encryptions of the same message m, (i.e. $c_i = [2^Z (r_i 2^{\beta-Z} + \bar{m})]^e \bmod n$ for $i = 1, 2$), the attacker evaluates:

$$\begin{aligned} \Delta &:= \frac{c_1 - c_2}{2^{eZ}\, 2^{\beta - Z}} \bmod n \\ &\equiv \underbrace{(r_1 - r_2)}_{:=\omega} \underbrace{\Big[\sum_{j=0}^{e-1} (r_1\, 2^{\beta-Z} + \bar{m})^{e-1-j} (r_2\, 2^{\beta-Z} + \bar{m})^j \Big]}_{:=v} \pmod n \ . \end{aligned} \tag{1}$$

The attack consists in the following: assuming that $r_1 > r_2$ and the number of zeroes Z to be large enough so that $0 < \omega\, v < n$, relation (1) holds over the integers, and $\omega = r_1 - r_2$ must divide Δ. Therefore, by extracting the small factors of Δ one expects to reconstruct a candidate for ω. The correct guess for ω will lead to the message m using the low-exponent attack described in [7].

Letting R the bit-size of random r' (the standard specifies $R \geq 64$), M the bit size of $\bar{m}$, and N the bit size of modulus n, the condition $w \cdot v < n$ is satisfied whenever:

$$eR + (e-1) \times (M + 10) < N \ . \tag{2}$$

With $N = R + M + Z + 24$, equation (2) is equivalent to:

$$(e-1)R + (e-2)M + 10e - 34 < Z \ .$$

2.1 Determining the factors of Δ smaller than a bound B

The first step of our attack consists in computing a set $\mathcal{D}$ of divisors of Δ by extracting the primes $\mathcal{P} = \{p_1, \ldots, p_i\}$ that divide Δ and are smaller than a bound B. If all the prime factors of ω are smaller than B (in this case, ω is said to be B-smooth), then $\omega \in \mathcal{D}$. Since only a partial factorization of Δ is required, only factoring methods which complexity relies on the size of the prime factors are of interest here. We briefly recall four of these: trial division, Pollard's ρ method, $p-1$ method and Lenstra's elliptic curve method (ECM) and express for each method the asymptotic complexity $C(p)$ of extracting a factor p from a number n.

Trial division method: Trial division by primes smaller than a bound B demands a complexity of $p + \log n$ for extracting p.

Pollard's ρ-method [4]: Let p be a factor of n. Pollard's ρ-method consists in iterating a polynomial with integer coefficients f (that is, computing $f(x) \bmod n$, $f(f(x)) \bmod n$, and so on) until a collision modulo p is found (i.e. $x \equiv x' \pmod p$). Then with high probability $\gcd(x - x'(\bmod n), n)$ yields p. The complexity of extracting a factor p is $\mathcal{O}(\sqrt{p})$. In practice, prime factors up to approximately 60 bits can be extracted in reasonable time (less than a few hours on a workstation).

$p-1$ method: If $p-1$ is B-smooth then $p-1$ divides the product $\ell(B)$ of all primes smaller than B. Since $a^{p-1} \bmod p = 1$, we have $a^{\ell(B)} \bmod p = 1$ and thus $\gcd(a^{\ell(B)} - 1 \bmod n, n)$ gives p.

Lenstra's elliptic curve method (ECM) [11]: ECM is a generalization of the $p-1$ factoring method. Briefly, a point P of a random elliptic curve $\mathcal{E}$ modulo n is generated. If $\#\mathcal{E}/(p)$ (i.e. the order of the curve modulo p) is B-smooth, then $[\ell(B)]P = \mathcal{O}$, the point at infinity. This means that an illegal inversion modulo n has occurred and p is revealed. ECM extracts a factor p of n in $\exp((\sqrt{2} + \mathrm{o}(1))\sqrt{\log p \log\log p})$ expected running time. In practice, prime factors up to 80 bits can be pulled out in reasonable time (less than a few hours on a workstation).

Traditionally, $\psi(x, y)$ denotes the number of integers $z \leq x$ such that z is smooth with respect to the bound y. The theorem that follows gives an estimate for $\psi(x, y)$.

Theorem 1 ([9]). *For any non-negative real u, we have:*

$$\lim_{x\to\infty} \psi(x, x^{1/u})/x = \rho(u)\,,$$

where $\rho(u)$ is the so-called Dickman's function and is defined as:

$$\rho(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1 \\ \rho(n) - \displaystyle\int_n^t \frac{\rho(v-1)}{v}\,dv & \text{if } n \leq t < n+1 \end{cases}.$$

Theorem 1 shows that a *uniformly distributed* random integer z between 1 and x is $x^{1/u}$-smooth with probability $\rho(u)$. However, the integers referred to in the sequel are not uniformly distributed. Consequently, the probability and complexity estimates must be considered to be heuristic.

The probability that ω is B-smooth is approximately $\rho(R/\log_2 B)$. Thus using two ciphertexts, the probability of finding all factors of ω is $\rho(R/\log_2 B)$. When using k ciphertexts, $k\times(k-1)/2$ paired combinations can be obtained. Assuming statistical independence between the factorization of the corresponding w, approximately

$$k = \sqrt{2/\rho(R/\log_2 B)}$$

ciphertexts are required to compute the factorization of at least one ω in complexity:

$$C(B)/\rho(R/\log_2 B) \ .$$

In practice, a factorization algorithm starts with trial division up to some bound B' (we took $B' = 15000$), then Pollard's ρ-method and the $p-1$ method are applied, and eventually the ECM. In Table 1 we give the running times obtained on a Pentium 233-MHz to extract a prime factor of size L bits with the ECM, using the arithmetic library MIRACL [12].

Table 1. Running times for extracting a prime factor of L bits using the ECM.

L	32	40	48	56	64	72
time in seconds	6	15	50	90	291	730

This clearly shows that for $R \leq 72$, the factors of ω can be recovered efficiently. For $R > 72$ we estimate in Table 2 the execution time and the number of required ciphertexts, when only factors up to 72 bits are to be extracted.

Table 2. Running time and approximate number of ciphertexts needed to recover the factorization of at least one ω.

L	128	160	192	224	256
time in seconds	1719	3440	7654	19010	51127
number of ciphertexts	3	4	5	8	12

2.2 Identifying the candidates for ω

From the previous section we obtain a set of primes $\mathcal{P} = \{p_1, \ldots, p_i\}$ dividing Δ, such that the primes dividing ω are in $\mathcal{P}$. From $\mathcal{P}$ we derive a set $\mathcal{D} = \{\Delta_j\}$ of divisors of Δ, which contains ω. Denoting by $d(k)$ the number of divisors of an integer k, the following theorem [10] provides an estimate of the number of divisors of a random integer. We say that an arithmetical function $f(k)$ is of the *average order* of $g(k)$ if

$$f(1) + f(2) + \ldots + f(k) \sim g(1) + \ldots + g(k) \ .$$

We state:

Theorem 2. *The average order of $d(k)$ is $\log k$. More precisely, we have:*

$$d(1) + d(2) + \cdots + d(k) = k \log k + (2\gamma - 1)k + \mathrm{O}(\sqrt{k}) \, ,$$

where γ is Euler's constant.

Theorem 2 shows that if Δ was uniformly distributed between 1 and n then its number of divisors and consequently the average number of candidates for ω would be roughly $\log n$. Since Δ is not uniformly distributed this only provides an heuristic argument to show that the average number of candidates for ω should be polynomially bounded by $\log n$.

In practice, not all divisors Δ_j need to be tested since only divisors of length close to or smaller than R are likely to be equal to ω. Moreover, from Eq. (1) and letting $\bar{m}_2 = r_2\, 2^{\beta - Z} + \bar{m}$, we have:

$$\begin{aligned}
\Delta &= \omega \sum_{j=0}^{e-1} (\omega\, 2^{\beta-Z} + \bar{m}_2)^{e-1-j}\, \bar{m}_2^j \\
&= \omega \sum_{j=0}^{e-1} \sum_{k=0}^{e-1-j} \binom{e-1-j}{k} (\omega\, 2^{\beta-Z})^{e-1-j-k}\, \bar{m}_2^{j+k} \\
&= \omega \sum_{h=0}^{e-1} \left[\sum_{i=0}^{h} \binom{e-1-i}{h-i} \right] (\omega\, 2^{\beta-Z})^{e-1-h}\, \bar{m}_2^h \,,
\end{aligned}$$

whence, noting that $\sum_{i=0}^{h} \binom{e-1-i}{h-i} \equiv 0 \pmod{e}$ for $1 \leq h \leq e-1$,

$$\Delta \equiv \omega\, (\omega\, 2^{\beta-Z})^{e-1} \pmod{e} \,.$$

In particular, when e is prime, this simplifies to

$$\Delta \equiv \omega^e\, 2^{(\beta-Z)(e-1)} \equiv \omega \pmod{e} \,.$$

This means that only a Δ_j satisfying $\Delta \equiv \Delta_j\, (\Delta_j\, 2^{\beta-Z})^{e-1} \pmod{e}$ (or $\Delta \equiv \Delta_j \pmod{e}$ if e is prime) is a valid candidate for ω.

2.3 Recovering m using the low-exponent RSA with related messages attack

The low-exponent attack on RSA with related messages described in [7] consists in the following: assume that two messages m_1, m_2 verify a known polynomial relation $\mathcal{P}$ of the form

$$m_2 = \mathcal{P}(m_1) \quad \text{with } \mathcal{P} \in \mathbb{Z}_n[z] \text{ and } \deg(\mathcal{P}) = \delta \,,$$

and suppose further that the two corresponding ciphertexts c_1 and c_2 are known. Then $z = m_1$ is a common root of polynomials $\mathcal{Q}_1, \mathcal{Q}_2 \in \mathbb{Z}_n[z]$ given by

$$\mathcal{Q}_1(z) = z^e - c_1 \quad \text{and} \quad \mathcal{Q}_2(z) = (\mathcal{P}(z))^e - c_2 \,,$$

so that with high probability one recovers m_1 by

$$\gcd(\mathcal{Q}_1, \mathcal{Q}_2) = z - m_1 \pmod{n} \,.$$

From the previous section we obtain a set of divisors Δ_j of Δ, among which one is equal to ω. Letting $m_1 = \text{PKCS}(m, r_1)$ and $m_2 = \text{PKCS}(m, r_2)$ we have:

$$c_1 = m_1^e \pmod n, \quad c_2 = m_2^e \pmod n, \quad \text{and } m_2 = m_1 - 2^\beta \omega \ .$$

For a divisor Δ_j of Δ, the attacker computes:

$$\mathcal{R}_j(z) = \gcd(z^e - c_1, (z - 2^\beta \Delta_j)^e - c_2) \ .$$

If $\Delta_j = \omega$ then, with high probability, $\mathcal{R}_j(z) = z - m_1 \pmod n$, which yields the value of message m, as announced.

3 Comparison with Coppersmith's Attacks on Low-exponent RSA

Coppersmith's method is based on the following theorem [6]:

Theorem 3 (Coppersmith). *Let $\mathcal{P} \in \mathbb{Z}_n[x]$ be a univariate polynomial of degree δ modulo an integer n of unknown factorization. Let X be the bound on the desired solution. If $X < \frac{1}{2} n^{1/\delta - \varepsilon}$, one can find all integers x_0 with $\mathcal{P}(x_0) = 0 \pmod n$ and $|x_0| \leq X$ in time polynomial in $(\log n, \delta, 1/\varepsilon)$.*

Corollary 1 (Coppersmith). *Under the same hypothesis and provided that $X < n^{1/\delta}$, one can find all integers x_0 such that $\mathcal{P}(x_0) = 0 \pmod n$ and $|x_0| \leq X$ in time polynomial in $(\log n, \delta)$*

Theorem 3 applies in the following situations:

Stereotyped messages: Assume that the plaintext m consists of a known part $B = 2^k b$ and an unknown part x. The ciphertext is $c = m^e = (B + x)^e \pmod n$. Using Theorem 3 with the polynomial $\mathcal{P}(x) = (B + x)^e - c$, one can recover x from c if $|x| < n^{1/e}$.

Random padding: Assume that two messages m and m' satisfy an affine relation $m' = m + r$ with a small but unknown r. From the RSA-encryptions of the two messages:

$$c = m^e \bmod n \quad \text{and} \quad c' = (m + r)^e \bmod n \,,$$

we eliminate m from the two above equations by taking their resultant, which gives a univariate polynomial in r modulo n of degree e^2. Thus, if $|r| < n^{1/e^2}$, r can be recovered, wherefrom we derive m as in Section 2.3.

In our case of interest, for a message ending with Z zeroes, the stereotyped messages attack works for $e(M + R) < N$ and the random padding attack works for $e^2 R < N$. Neglecting constant terms, our method of Section 2 is effective for

$$eR + (e - 1)M < N \ .$$

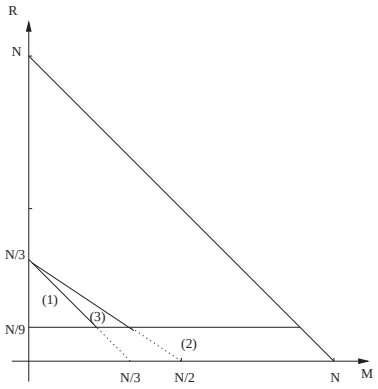


Fig. 1. Domains of validity for $e = 3$ of Coppersmith's stereotyped attack (1), Coppersmith's random padding attack (2) and our attack (3).

Consequently, as illustrated in Figure 1, for $e = 3$, our method improves Coppersmith's method whenever

$$\begin{cases} \dfrac{N}{e^2} < R < \dfrac{N}{e} \quad \text{and} \\ \dfrac{N}{e} - R < M < \dfrac{N}{e-1} - \dfrac{e}{e-1}R \end{cases} .$$

4 A Chosen Plaintext Attack for Arbitrary Exponents

4.1 Description

In this section we describe a chosen plaintext attack against PKCS#1 v1.5 encryption for an arbitrary exponent e. The attack makes use of a known flaw in El Gamal encryption [3] and works for very short messages only. As in Section 2 we only consider messages ending by Z zeroes:

$$m = \bar{m} \| \mathtt{0} \ldots \mathtt{0}_2 \ .$$

For a random r' consisting of nonzero bytes, the message m is transformed using PKCS#1 v1.5 into:

$$\text{PKCS}(m, r') = \mathtt{0002}_{16} \| r' \| \mathtt{00}_{16} \| \bar{m} \| \mathtt{0} \ldots \mathtt{0}_2$$

and encrypted into $c = \text{PKCS}(m, r')^e \bmod n$. Letting $x = \mathtt{0002}_{16} \| r' \| \mathtt{00}_{16} \| \bar{m}$, we can write

$$\text{PKCS}(m, r') = x \, 2^Z \ .$$

We define $y = c/2^{eZ} = x^e \pmod n$, M the bit-size of $\bar{m}$, and X the bit-size of x. Hence, we have $X = M + R + 10$. Assuming that $x = x_1\, x_2$ where x_1 and x_2 are integers smaller than a bound B, we construct the table:

$$\frac{y}{i^e} \bmod n \quad \text{for } i = 1, \ldots, B$$

and for each $j = 0, \ldots, B$ we check whether $j^e \bmod n$ belongs to the table, in which case we have $y/i^e = j^e \bmod n$. Hence, from $\{i, j\}$ we recover $x = i \cdot j$, which leads to the message m.

4.2 Analysis

The attack requires $\mathcal{O}\big(B(\log n)((\log n)^3 + \log B)\big)$ operations. Let $\phi(x, y)$ denote the number of integers $v < x$ such that v can be written as $v = v_1\, v_2$ with $v_1 < y$ and $v_2 < y$. The following theorem gives a lower bound for $\phi(x, y)$.

Theorem 4. *For $x \to \infty$ and $1/2 < \alpha < 1$,*

$$\liminf \phi(x, x^\alpha)/x \geq \log \frac{\alpha}{1-\alpha} \ . \tag{3}$$

Proof. For $y > \lceil \sqrt{x} \rceil$, we note:

$$\mathcal{T}(x, y) = \{v < x, \text{such that } v \text{ is } y\text{-smooth and not } \lceil x/y \rceil\text{-smooth}\} \ .$$

Any integer $v \in \mathcal{T}(x, y)$ has a prime factor p standing between $\lceil x/y \rceil$ and y, and so $v = p\, r$ with $p < y$ and $r < y$. Consequently,

$$\phi(x, y) \geq \#\mathcal{T}(x, y) \ . \tag{4}$$

From Theorem 1 and $\rho(t) = 1 - \log t$ for $1 \leq t \leq 2$, we have:

$$\lim_{x \to \infty} \#\mathcal{T}(x, x^\alpha)/x = \log \frac{\alpha}{1-\alpha} \, ,$$

which, using Eq. (4) gives (3). □

Since x is not uniformly distributed between zero and 2^X, Theorem 4 only provides a heuristic argument to show that when taking $B = 2^{\alpha X}$ with $\alpha > 1/2$, then with probability greater than

$$\log \frac{\alpha}{1-\alpha} \, ,$$

the attack recovers x in complexity $2^{\alpha X + o(1)}$.

Thus, an eight-bit message encrypted with PKCS#1 v1.5 with a 64-bit random padding string can be recovered with probability $\simeq 0.16$ in time and space complexity approximately 2^{44} (with $\alpha = 0.54$).

5 Experiments and Counter-measures

A number of counter-measures against Bleichenbacher's attack are listed on RSA Laboratories' web site (`http://www.rsa.com/rsalabs/`). A first recommendation is a rigorous format check of all decrypted messages. This has no effect on our attack since we never ask the legitimate receiver to decrypt anything. A second quick fix consists in asking the sender to demonstrate knowledge of m to the recipient which is done by disclosing some additional piece of information. This also has no effect on our attack. The same is true for the third correction, where a hash value is incorporated in m, if the hash value occupies the most significant part of the plaintext i.e.

$$\text{PKCS}(m, r') = \mathtt{0002}_{16} \| r' \| \mathtt{00}_{16} \| \, \text{SHA}(m) \| m \ .$$

A good way to thwart our attack is to limit Z. This can be very simply achieved by forcing a constant pattern τ in $\text{PKCS}(m, r')$:

$$\text{PKCS}(m, r') = \mathtt{0002}_{16} \| r' \| \mathtt{00}_{16} \| m \| \tau \ .$$

This presents the advantage of preserving compatibility with PKCS#1 v1.5 and being very simple to implement. Unfortunately, the resulting format is insufficiently protected against [2]. Instead, we suggest to use:

$$\text{PKCS}(m, r') = \mathtt{0002}_{16} \| r' \| \mathtt{00}_{16} \| m \| \, \text{SHA}(m, r') \, ,$$

which appears to be an acceptable short-term choice (r' was added in the hash function to better resist [2] at virtually no additional cost). For long-term permanent solutions, we recommend OAEP (PKCS#1 v2.0) [1].

6 Extensions and Conclusions

We proposed two new chosen-plaintext attacks on the PKCS#1 v1.5 encryption standard. The first attack applies to small public exponents and shows how messages ending by sufficiently many zeroes can be recovered from the ciphertexts corresponding to the same plaintext. It is worth seeing our technique as a cryptanalytic tool of independent interest, which provides an extension of Coppersmith's low-exponent attack. Our second attack, although remaining of exponential complexity in a strict sense, shows how to extend the weakness to any public exponent in a practical way.

The attacks can, of course, be generalized in several ways. For instance, one can show that the padding format:

$$\mu(m_1, m_2, r') = \mathtt{0002}_{16} \| m_1 \| r' \| \mathtt{00}_{16} \| m_2$$

(where the plaintext $m = m_1 \| m_2$ is spread between two different locations), is equally vulnerable to the new attack: re-defining $r'' = m_1 \| r'$, we can run the

attack (as is) on $\text{pkcs}(m, r'')$ and notice that the size of ω will still be R' given that the most significant part of r'' is always constant.

We believe that such examples illustrate the risk induced by the choice of *ad hoc* low-cost treatments as message paddings, and highlights the need for carefully scrutinized encryption designs, strongly motivating (once again) the search for provably secure encryption schemes.

Acknowledgements

We wish to thank the referees for their valuable remarks and improvements to this work.

References

1. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Advances in Cryptology – EUROCRYPT '94, vol. 950 of Lecture Notes in Computer Science, pp. 92–111, Springer-Verlag, 1994.
2. D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1*, Advances in Cryptology – CRYPTO '98, vol. 1462 of Lecture Notes in Computer Science, pp. 1–12, Springer-Verlag, 1998.
3. D. Boneh, Personal communication.
4. R. Brent, *An improved Monte Carlo factorization algorithm*, Nordisk Tidskrift för Informationsbehandling (BIT), vol. 20, pp. 176–184, 1980.
5. D. Coppersmith, *Finding a small root of a univariate modular equation*, Advances in Cryptology – EUROCRYPT '96, vol. 1070 of Lecture Notes in Computer Science, pp. 155–165, Springer-Verlag, 1996.
6. D. Coppersmith, *Small solutions to polynomial equations, and low exponent RSA vulnerabilities*, J. of Cryptology, 10(4), pp. 233-260, 1997.
7. D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, *Low exponent RSA with related messages*, Advances in Cryptology – EUROCRYPT '96, vol. 1070 of Lecture Notes in Computer Science, pp. 1–9, Springer-Verlag, 1996.
8. Y. Desmedt and A. Odlyzko. *A chosen text attack on the* RSA *cryptosystem and some discrete logarithm schemes*, Advances in Cryptology – CRYPTO '85, vol. 218 of Lecture Notes in Computer Science, pp. 516–522, Springer-Verlag, 1986.
9. K. Dickman, *On the frequency of numbers containing prime factors of a certain relative magnitude*, Arkiv för matematik, astronomi och fysik, vol. 22A, no. 10, pp. 1–14, 1930.
10. G.H. Hardy and E.M. Wright, *An Introduction to the theory of numbers*, Fifth edition, Oxford University Press, 1979.
11. H. Lenstra, *Factoring integers with elliptic curves*, Annals of mathematics 126, 1987.
12. Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL), available at `ftp://ftp.compapp.dcu.ie/pub/crypto/miracl.zip`.
13. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, vol. 21-2, pp. 120-126, 1978.
14. RSA Data Security, PKCS #1: *RSA Encryption Standard*, Nov. 1993, Version 1.5.
15. RSA Laboratories, PKCS #1: *RSA cryptography specifications*, Sep. 1998, Version 2.0.

A A Full-scale 1024-bit Attack

To confirm the validity of our attack, we experimented it on RSA Laboratories' official 1024-bit challenge RSA-309 for the public exponent $e = 3$. As a proof of proper generation r'_1 and r'_2 were chosen to be RSA-100 mod 2^{128} and RSA-110 mod 2^{128}. The parameters are $N = 1024$, $M = 280$, $R = 128$, $Z = 592$ and $\beta = 880$. Note that since $R > N/9$ and $R+M > N/3$, Coppersmith's attack on low-exponent RSA does not apply here.

$$
\begin{aligned}
n &= \text{RSA-309} \\
&= \texttt{bdd14965 645e9e42 e7f658c6 fc3e4c73 c69dc246 451c714e b182305b 0fd6ed47} \\
&\quad \texttt{d84bc9a6 10172fb5 6dae2f89 fa40e7c9 521ec3f9 7ea12ff7 c3248181 ceba33b5} \\
&\quad \texttt{5212378b 579ae662 7bcc0821 30955234 e5b26a3e 425bc125 4326173d 5f4e25a6} \\
&\quad \texttt{d2e172fe 62d81ced 2c9f362b 982f3065 0881ce46 b7d52f14 885eecf9 03076ca5} \\
r'_1 &= \text{RSA-100} \bmod 2^{128} \\
&= \texttt{f66489d1 55dc0b77 1c7a50ef 7c5e58fb} \\
r'_2 &= \text{RSA-110} \bmod 2^{1}28 \\
&= \texttt{e2a5a57d e621eec5 b14ff581 a6368e9b} \\
m &= \bar{m}\, 2^Z \\
&\quad \texttt{I 'm a ciphertext, please break} \\
&\quad \texttt{0049276d 20612063 69706865 72746578 742c2070 6c656173 65206272 65616b20} \\
&\quad \texttt{m e !} \\
&\quad \texttt{6d652021}
\end{aligned}
$$

$$
\begin{aligned}
\mu_1 &= \text{PKCS}(m, r'_1) \\
&= \texttt{0002f664 89d155dc 0b771c7a 50ef7c5e 58fb0049 276d2061 20636970 68657274} \\
&\quad \texttt{6578742c 20706c65 61736520 62726561 6b206d65 20210000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000}
\end{aligned}
$$

$$
\begin{aligned}
\mu_2 &= \text{PKCS}(m, r'_2) \\
&\quad \texttt{0002e2a5 a57de621 eec5b14f f581a636 8e9b0049 276d2061 20636970 68657274} \\
&\quad \texttt{6578742c 20706c65 61736520 62726561 6b206d65 20210000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000}
\end{aligned}
$$

$$
\begin{aligned}
c_1 &= \mu_1^3 \bmod n \\
&= \texttt{2c488b6f cf2e3d4c 01b82776 64790af0 d78f82fd 4605fda2 76b9356d 80e82cfb} \\
&\quad \texttt{8737340f 5a7091b0 38c4bb41 ae6462d9 f751766c c343c87b 54397ca2 647d6a81} \\
&\quad \texttt{3609d876 f29554e0 9efcbf2d b49d8300 5fce9ea8 80fd9cf2 476fbab0 257f1462} \\
&\quad \texttt{d295a4cb 5468bb86 b3151a49 14e51ed1 7cbc083c 9ae0b4da 9c2a7de0 079df4a0}
\end{aligned}
$$

$$\begin{aligned} c_2 &= \mu_2^3 \bmod n \\ &= \texttt{829da9a7 af2c61ed 7bb16f94 7cb90aa7 df8b99df c06017d7 3afc80fd 64494abb} \\ &\quad \texttt{3c1cb8db 1167eccd d1b6d09e 8ca5a98c c5e19620 b6313eef 495169d7 9ed9a2b1} \\ &\quad \texttt{cb393e7d 45bea586 49e20986 9a2399f7 f70dd819 90183e1a 3c6a971a 33497e57} \\ &\quad \texttt{f0ad9fb9 0c7d331e 7108d661 4c487a85 36cf7750 060811d8 70b8a040 e0c39999} \end{aligned}$$

Using the ECM it took a few hours on a single workstation to find that:

$$\Delta = p_1^5 \times \prod_{i=2}^{10} p_i$$

where all the p_i are primes. Amongst the $3072 = 6 \times 2^9$ possible divisors only 663 corresponded to 128-bit candidates $\{\Delta_1, \Delta_2, \ldots, \Delta_{663}\}$ where the Δ_i are in decreasing order. Then we computed:

$$\mathcal{R}_j(z) = \gcd(z^e - c_1, (z - 2^\beta \Delta_j)^e - c_2) \quad \text{for } 1 \le j \le 663 \ .$$

For $j \neq 25$, $\mathcal{R}_j(z) = 1$ and for $j = 25$ we obtained:

$$\mathcal{R}_{25}(z) = z - m_1 \ .$$

One can check that:

$$\Delta_{25} = w = p_1^5 \, p_2 \, p_3 \, p_4 \, p_5 \, p_8 \, ,$$

and

$$m_1 = \mu_1 = \text{PKCS}(m, r_1') \ .$$

$$\begin{aligned} \Delta &= \texttt{00000001 fa75bf4e 390bdf4b 7a0524e0 b9ebed20 5758be2e f1685067 1de199af} \\ &\quad \texttt{0f8714f7 077a6c47 6870ea6d 2de9e7fb 3c40b8d2 017c0197 f9533ed1 f4fe3eab} \\ &\quad \texttt{836b6242 aa03181a 56a78001 7c164f7a c54ecfa7 73583ad8 ffeb3a78 eb8bcbe2} \\ &\quad \texttt{8869da15 60be7922 699dc29a 52038f7b 83e73d4e 7082700d 85d3a720} \\ p_1 &= \texttt{00000002}, \ p_2 = \texttt{00000007}, \ p_3 = \texttt{00000035}, \ p_4 = \texttt{000000c5}, \ p_5 = \texttt{4330e379} \\ p_6 &= \texttt{548063d7}, p_7 = \texttt{001ebf96 ff071021}, \ p_8 = \texttt{0000021b ac4d83ae 7dedba55} \\ p_9 &= \texttt{0000128a ec52c6ec 096996bf} \\ p_{10} &= \texttt{00000022 e3b1a6b0 13829b67 f604074a 5a1135b3 45be0835 ea407ed7 8138a27a} \\ &\quad \texttt{112e78c8 131f3bc3 b6d17dc0 e8a905f1 ca4b6aff 680bc58c 4962309d c7aaccad} \\ &\quad \texttt{2116235c b0d6803e e0a58ca7 55cbea23 e936f189 a76dfbeb} \\ \Delta_{25} &= \texttt{13bee453 6fba1cb1 6b2a5b6d d627ca60} \\ \mathcal{R}_{25}(z) &= z - m_1 \\ m_1/2^Z \bmod 2^M &\quad \underset{\texttt{I}}{\texttt{0049276d}} \ \underset{\texttt{'}}{\texttt{20612063}} \ \underset{\texttt{m}}{\texttt{69706865}} \ \underset{\texttt{a}}{\texttt{72746578}} \ \underset{\texttt{ }}{\texttt{742c2070}} \ \underset{\texttt{c}}{\texttt{6c656173}} \ \underset{\texttt{i}}{\texttt{65206272}} \ \underset{\texttt{p}}{\texttt{65616b20}} \\ &\quad \underset{\texttt{h}}{\texttt{6d652021}} \ \underset{\texttt{e}}{} \ \underset{\texttt{e}}{} \ \underset{\texttt{r}}{} \ \underset{\texttt{t}}{} \ \underset{\texttt{e}}{} \ \underset{\texttt{x}}{} \ \underset{\texttt{t}}{} \ \underset{\texttt{,}}{} \ \underset{\texttt{ }}{} \ \underset{\texttt{p}}{} \ \underset{\texttt{l}}{} \ \underset{\texttt{e}}{} \ \underset{\texttt{a}}{} \ \underset{\texttt{s}}{} \ \underset{\texttt{e}}{} \ \underset{\texttt{ }}{} \ \underset{\texttt{b}}{} \ \underset{\texttt{r}}{} \ \underset{\texttt{e}}{} \ \underset{\texttt{a}}{} \ \underset{\texttt{k}}{} \\ &\quad \texttt{6d652021} \end{aligned}$$

On Rabin-type Signatures*

[Published in B. Honary, Ed., *Cryptography and Coding*, vol. 2260 of *Lecture Notes in Computer Science*, pp. 99–113, Springer-Verlag, 2001.]

Marc Joye[1] and Jean-Jacques Quisquater[2]

[1] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos Cedex, France
marc.joye@gemplus.com
http://www.geocities.com/MarcJoye/
[2] UCL Crypto Group, Université catholique de Louvain
Place du Levant 3, 1348 Louvain-la-Neuve, Belgium
jjq@dice.ucl.ac.be
http://www.uclcrypto.org/

Abstract. This paper specializes the signature forgery by Coron, Naccache and Stern (1999) to Rabin-type systems. We present a variation in which the adversary may derive the private keys and thereby forge the signature on *any* chosen message. Further, we demonstrate that, contrary to the RSA, the use of larger (even) public exponents does not reduce the complexity of the forgery. Finally, we show that our technique is very general and applies to any Rabin-type system designed in a unique factorization domain, including the Williams' M^3 scheme (1986), the cubic schemes of Loxton et al. (1992) and of Scheidler (1998), and the cyclotomic schemes (1995).

Keywords. Rabin-type systems, digital signatures, signature forgeries, factorization.

1 Introduction

In this paper, we specialize the signature forgery of Coron, Naccache and Stern [8] to Rabin-type systems [19, 24]. We present a variation in which the adversary may derive the private keys and thereby forge the signature on *any* chosen message. Further, we demonstrate that, contrary to the RSA, systems using larger (even) public exponents are *equally* susceptible to the presented forgery. We also show that our technique is very general and applies to any Rabin-type systems designed in a unique factorization domain, including the Williams' M^3 scheme [27], the cubic schemes of Loxton et al. [16] and of Scheidler [20], and the cyclotomic schemes [21].

* A working draft of this work was presented at the ISO/IEC JTC1/SC27/WG2 meeting in August 1999.

As an application, we analyze the implications of our forgery against the PKCS #1 standard [4]. Finally, and of independent interest, we propose a *generic* technique (i.e., applicable to *any* encoding message method) that reduces the overall complexity of a forgery from n to $\sqrt{n}$.

The rest of this paper is organized as follows. We first begin by a brief presentation of Rabin-type systems. Next, in Section 3, we review the Coron-Naccache-Stern forgery and turn it into an universal forgery against Rabin-type signature schemes. In Section 4, we generalize the forgery to higher exponents and higher degree schemes. We apply it to PKCS #1 encoding method in Section 5. We also present a generic technique for reducing the complexity of the forgery. Finally, we conclude in Section 6.

2 Rabin-type Systems

In this section, following the IEEE/P1363 specifications for public-key cryptography [2] (see also [17, Chapter 11]), we present a modified version of the Rabin-Williams signature scheme [19, 24]. The scheme consists of three algorithms: the `setup`, the `signature` and the `verification`. For setting up the system, each user generates a pair of public/private keys. The private key is used to sign messages with the signature algorithm. Using the corresponding public key, a signature can then be verified and the signed message recovered with the verification algorithm.

`setup`: Generate two primes p, q such that $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$ and compute $n = pq$. Define an appropriate "representation" function $R : \mathcal{M} \to \mathcal{M}_R : m \mapsto \widetilde{m} = R(m)$, where $\mathcal{M}$ is the set of valid messages and $\mathcal{M}_R = \{\widetilde{m} = R(m) \in (\mathbb{Z}/n\mathbb{Z})^* : \widetilde{m} \equiv 6 \pmod{16}\}$ is the set of message representatives. The public key is n and the private key is $d = (n-p-q+5)/8$.

`signature`: Compute $\widetilde{m} = R(m)$ and $\widehat{m}$ given by

$$\widehat{m} = \begin{cases} \widetilde{m} \bmod n & \text{if } (\widetilde{m}|n) = 1 \\ \widetilde{m}/2 \bmod n & \text{if } (\widetilde{m}|n) = -1 \end{cases} .$$

The signature on message m is $s = \widehat{m}^d \bmod n$.

`verification`: Compute $m' = s^2 \bmod n$. Then, take

$$\widetilde{m} = \begin{cases} m' & \text{if } m' \equiv 6 \pmod 8 \\ 2m' & \text{if } m' \equiv 3 \pmod 8 \\ n - m' & \text{if } m' \equiv 7 \pmod 8 \\ 2(n - m') & \text{if } m' \equiv 2 \pmod 8 \end{cases} .$$

If $\widetilde{m} \in \mathcal{M}_R$ then the signature is accepted and message m is recovered from $\widetilde{m}$.

Remark 1. A signature scheme with appendix can also be defined along these lines. In that case, the set of messages representatives is given by $\mathcal{M}_R = \{\widetilde{m} \in (\mathbb{Z}/n\mathbb{Z})^* : \widetilde{m} \equiv 10 \pmod{16}\}$.

Remark 2. The recommended function R for message encoding is the international standard ISO/IEC 9796-1 [3]. Another possible encoding is specified in PKCS #1 v2.0 [4]; this latter, however, only covers signature schemes with appendix.

Remark 3. The presented scheme supposes $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$. Similar schemes with form-free primes may be found in [26, 13].

Noticing that $2d = (p-1)(q-1)/4+1$, the correctness of the method follows from the next lemma; moreover, it uses the fact that if $n \equiv 5 \pmod 8$ then $(-z|n) = (z|n) = -(2z|n)$.

Lemma 1. *Let $n = pq$, where p, q are distinct primes and $p, q \equiv 3 \pmod 4$. If $(z|n) = 1$, then*

$$z^{(p-1)(q-1)/4} \equiv \pm 1 \pmod n \ .$$

Proof. See [24, Lemma 1]. □

3 Signature Forgeries

This section reviews the Coron-Naccache-Stern forgery [8] when applied to the Rabin-Williams scheme. In the second part, we modify it into an universal forgery so that the signature on any message can be obtained without knowing the private key.

3.1 Coron-Naccache-Stern forgery

As aforementioned, for each message representative $\widetilde{m}_i$, $\widehat{m}_i = \widetilde{m}_i$ if $(\widetilde{m}_i|n) = 1$ and $\widehat{m}_i = \widetilde{m}_i/2$ if $(\widetilde{m}_i|n) = -1$; the corresponding signature is then given by $s_i = {\widehat{m}_i}^d \bmod n$. (Note here that $(\widehat{m}_i|n) = 1$.)

Suppose that an adversary has collected several pairs $(\widehat{m}_i, s_i)$ such that the $\widehat{m}_i$'s are smooth (modulo n). More precisely, suppose she knows

$$\widehat{m}_i \equiv (-1)^{v_{0,i}} \prod_{1 \le j \le B} {p_j}^{v_{j,i}} \pmod n \,, \tag{1}$$

where $p_1 < \cdots < p_B$ are prime and $v_{j,i} \in \mathbb{Z}$, and $s_i = {\widehat{m}_i}^d \bmod n$, for $1 \le i \le \ell$. Then she can forge the signature on a message m_τ, provided that $\widehat{m}_\tau$ is smooth (modulo n), as follows.[1,2]

[1] It is here essential to note that the smoothness requirement must only be satisfied modulo n. For example, 197 is prime in $\mathbb{Z}$ but is 3-smooth as an element of $(\mathbb{Z}/437\mathbb{Z})$, i.e., $197 \equiv 2^9 \cdot 3^{-3} \pmod{437}$.

[2] In [8], the authors only consider positive "messages" $\widehat{m}_i$. We slightly generalize their presentation by introducing the term $(-1)^{v_{0,i}}$ in Eq. (1).

To each $\widehat{m}_i$, she associates the B-tuple $\vec{V}_i$ given by $\vec{V}_i = (v_{1,i}, \ldots, v_{B,i})$. Let $\vec{V}_\tau$ denote the B-tuple corresponding to $\widehat{m}_\tau$. If there exist integers β_i such that $\vec{V}_\tau \equiv \sum_{1 \le i \le \ell} \beta_i \vec{V}_i \pmod 4$, then

$$v_{j,\tau} = \sum_{1 \le i \le \ell} \beta_i \, v_{j,i} - 4\gamma_j \quad (1 \le j \le B) \tag{2}$$

for some $\gamma_j \in \mathbb{Z}$.
Hence, the signature on message m_τ can be expressed as

$$\begin{aligned} s_\tau := {\widehat{m}_\tau}^d \bmod n &\equiv \Big[(-1)^{v_{0,\tau}} \prod_{1 \le j \le B} {p_j}^{v_{j,\tau}}\Big]^d \qquad \text{(from Eq. (1))} \\ &\equiv \prod_{1 \le j \le B} {p_j}^{v_{\tau,j}\, d} \qquad \text{(since } d \text{ is even)} \\ &\equiv \prod_{1 \le i \le \ell} \prod_{1 \le j \le B} {p_j}^{(\beta_i v_{j,i} - 4\gamma_j) d} \qquad \text{(from Eq. (2))} \\ &\equiv \prod_{1 \le i \le \ell} {s_i}^{\beta_i} \prod_{1 \le j \le B} {p_j}^{-2\gamma_j} \pmod n \ . \end{aligned} \tag{3}$$

The only difference with [8] is that we use the weaker relation ${p_j}^{4d} \equiv {p_j}^2 \pmod n$. (The relation ${p_j}^{2d} \equiv \pm p_j \pmod n$ only holds when $(p_j|n) = 1$; see Lemma 1.)

3.2 Universal forgery

We will now show that we can do much better than forging the signature on a smooth $\widehat{m}_\tau$, namely, forging the signature on *any* chosen $\widehat{m}_\tau$. We need the following proposition.

Proposition 1. *Let $n = pq$, where p, q are distinct primes and $p, q \equiv 3 \pmod 4$ and let $d = (n - p - q + 5)/8$. If $(z|n) = -1$, then*

$$\gcd\big({z^{2d} \mp z \pmod n}, n\big) = p \text{ or } q \ .$$

Proof. Since $p, q \equiv 3 \pmod 4$, both $(p-1)/2$ and $(q-1)/2$ are odd. Hence, $z^{2d} \equiv z^{(p-1)(q-1)/4}\, z \equiv (z|p)^{(q-1)/2}\, z \equiv (z|p)\, z \pmod p$, and similarly $z^{2d} \equiv (z|q)\, z \pmod q$. Noting that $(z|p) = -(z|q) = \pm 1$, the proposition is proved. □

The above proposition suggests that if an adversary can derive the signature on an $\widehat{m}_\tau$ such that $(\widehat{m}_\tau|n) = -1$, then she can factor the modulus by computing $\gcd\big({s_\tau}^2 - \widehat{m}_\tau \pmod n, n\big)$. This, however, is *not* possible. Define

$$\mathcal{J}(n, B) = \Big\{1 \le j \le B : p_j \text{ is prime and } \Big(\frac{p_j}{n}\Big) = -1\Big\} \ . \tag{4}$$

Since $(\widehat{m}_i|n) = 1$ (cf. beginning of § 3.1), we must have

$$\sum_{j \in \mathcal{J}(n,B)} v_{j,i} \equiv 0 \pmod 2 \ . \tag{5}$$

So, any linear combination, as done in Eq. (2), will always yield $\sum_{j \in \mathcal{J}(n,B)} v_{j,\tau} \equiv 0 \pmod 2$, resulting in $(\widehat{m}_\tau|n) = 1$.

But there is another way to consider Proposition 1: If the adversary is able to obtain the signature on $\widehat{m}_\tau = \pm \widehat{r}_\tau{}^2$ for some $\widehat{r}_\tau$ such that $(\widehat{r}_\tau|n) = -1$ (note here that $(\widehat{m}_\tau|n) = (\pm 1|n) \cdot (\widehat{r}_\tau{}^2|n) = 1 \cdot 1 = 1$), then

$$\gcd\big(s_\tau - \widehat{r}_\tau \pmod n, n\big) \tag{6}$$

will give a non-trivial factor of n. To make this feasible, in addition to verify Eq. (2), the components of $\vec{V}_\tau$ must satisfy

$$v_{j,\tau} \equiv 0 \pmod 2 \quad \text{for all } 1 \le j \le B \tag{7}$$

and

$$\sum_{j \in \mathcal{J}(n,B)} v_{j,\tau} \equiv 2 \pmod 4 \ . \tag{8}$$

The first condition ensures that the "message" corresponding to $\vec{V}_\tau$ is a square (i.e., $\widehat{m}_\tau = \pm \widehat{r}_\tau{}^2$), while the second condition ensures that $(\widehat{r}_\tau|n) = -1$. Replacing $v_{j,\tau}$ by Eq. (2), Eqs (7) and (8) can respectively be rewritten as

$$\sum_{1 \le i \le \ell} \beta_i \, v_{j,i} \equiv 0 \pmod 2 \quad \text{for all } 1 \le j \le B \tag{9}$$

and, defining $2\zeta_i = (\sum_{j \in \mathcal{J}(n,B)} v_{j,i}) \bmod 4$ ($\in \{0, 2\}$ from Eq. (5)),

$$\begin{aligned} &\sum_{j \in \mathcal{J}(n,B)} \sum_{1 \le i \le \ell} \beta_i \, v_{j,i} \equiv \sum_{1 \le i \le \ell} \beta_i \sum_{j \in \mathcal{J}(n,B)} v_{j,i} \equiv \sum_{1 \le i \le \ell} \beta_i \, 2\zeta_i \equiv 2 \pmod 4 \\ &\iff \sum_{1 \le i \le \ell} \beta_i \, \zeta_i \equiv 1 \pmod 2 \ . \end{aligned} \tag{10}$$

To sum up, an adversary can recover the secret factorization of n by carrying out the following steps:

(I) For $1 \le i \le \ell$, write $\widehat{m}_i \equiv (-1)^{v_{0,j}} \prod_{1 \le j \le B} {p_j}^{v_{j,i}} \pmod n$;

(II) Define $\vec{V}_i = (v_{1,i}, \ldots, v_{B,i})$ and $\zeta_i = \dfrac{(\sum_{j \in \mathcal{J}(n,B)} v_{j,i}) \bmod 4}{2}$;

(III) Find $\beta_1, \ldots, \beta_\ell$ such that

(a) for $1 \le j \le B$, $\sum_{1 \le i \le \ell} \beta_i \, v_{j,i} \equiv 0 \pmod 2$;

(b) $\displaystyle\sum_{1\le i\le \ell} \beta_i\,\zeta_i \equiv 1 \pmod 2$;

(IV) Compute $\vec{V}_\tau = (v_{1,\tau},\ldots,v_{B,\tau}) \in (\mathbb{Z}/4\mathbb{Z})^B$,
where $v_{j,\tau} = \displaystyle\sum_{1\le i\le \ell} \beta_i\, v_{j,i} - 4\gamma_j$ for some $\gamma_j \in \mathbb{Z}$;

(V) Set $\widehat{r}_\tau = \displaystyle\prod_{1\le j\le B} {p_j}^{v_{j,\tau}/2} \pmod n$;

(VI) Compute $s_\tau = \displaystyle\prod_{1\le i\le \ell} {s_i}^{\beta_i} \prod_{1\le j\le B} {p_j}^{-2\gamma_j} \bmod n$;

(VII) Recover the factors of n by computing $\gcd(s_\tau - \widehat{r}_\tau \pmod n, n)$.

Example 1. Here is a "toy" example to illustrate the forgery. Let $p = 8731$ ($\equiv 3 \pmod 8$) and $q = 3079$ ($\equiv 7 \pmod 8$) yielding a modulus $n = 26882749$. So, the private exponent is given by $d = 3358868$.
We consider p_4-smooth message representatives $\widetilde{m}_i \in \mathcal{M}_R$. We have $\mathcal{J}(n,4) = \{2,7\}$. Suppose, we are given:

$\widetilde{m}_i$	$\widehat{m}_i \pmod n$	s_i	$\vec{V}_i$	ζ_i
70	70 ($= 2\cdot 5\cdot 7$)	8417525	$(1,0,1,1)$	1
294	147 ($= 3\cdot 7^2$)	11480098	$(0,1,0,2)$	1
486	243 ($= 3^5$)	16287310	$(0,5,0,0)$	0
630	630 ($= 2\cdot 3^2\cdot 5\cdot 7$)	1630174	$(1,2,1,1)$	1

Conditions (III-a) and (III-b) yield

$$\begin{cases} \beta_1 + \beta_4 \equiv 0 \pmod 2 \\ \beta_2 + \beta_3 \equiv 0 \pmod 2 \\ \beta_1 + \beta_2 + \beta_4 \equiv 1 \pmod 2 \end{cases} \iff \begin{cases} \beta_1 \equiv \beta_4 \pmod 2 \\ \beta_2 \equiv \beta_3 \equiv 1 \pmod 2 \end{cases} .$$

Taking $\beta_1 = \beta_4 = 0$ and $\beta_2 = \beta_3 = 1$, we have $\vec{V}_\tau = (0,2,0,2)$, which corresponds to $\widehat{m}_\tau = 3^2\cdot 7^2 = 21^2$ whose signature is given by $s_\tau = {s_2}^1\, {s_3}^1\, 3^{-2} \bmod n = 8076196$. So, the factorization of n is obtained by computing $\gcd(8076196 - 21, n) = 8731$ ($= p$). ♢

We will see below (Algorithm 1) a simple method to compute a solution $(\beta_1,\ldots,\beta_\ell) \in (\mathbb{Z}/2\mathbb{Z})^\ell$. This is a slight modification of Algorithm 2.3.1 in [7, pp. 56–57] (see also [12, Algorithm N, pp. 425–426]).

Using matrix notations, Conditions (III-a) and (III-b) can be rewritten as

$$\underbrace{\begin{pmatrix} v_{1,1} & \cdots & v_{1,\ell} & 0 \\ \vdots & & \vdots & \vdots \\ v_{B,1} & \cdots & v_{B,\ell} & 0 \\ \zeta_1 & \cdots & \zeta_\ell & 1 \end{pmatrix}}_{\substack{(\bmod 2) \\ :=\mathbf{U}}} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_\ell \\ 1 \end{pmatrix} \equiv \vec{0} \pmod 2 . \tag{11}$$

So, the problem is reduced to find a vector $\vec{K} = (\kappa_1, \ldots, \kappa_\ell, \kappa_{\ell+1}) \in \ker \mathbf{U}$ (i.e., the kernel of matrix $\mathbf{U}$), whose last coordinate, $\kappa_{\ell+1}$, is equal to 1. In that case, we have $\beta_i = \kappa_i$, $1 \leq i \leq \ell$.

Algorithm 1. This algorithm computes a solution (if any) $\beta_1, \ldots, \beta_\ell$ satisfying Eq. (11). We let $u_{r,c}$ denote the entry (modulo 2) at row r and column c of matrix $\mathbf{U}$.

1. [initialization] Set $c \leftarrow 1$ and for $1 \leq j \leq B+1$, set $t_j \leftarrow 0$.
2. [scanning] If there is some r in the range $1 \leq r \leq B+1$ such that $u_{r,c} = 1$ and $t_r = 0$, then go to Step 3. Otherwise, go to Step 5.
3. [elimination] For all $j \neq r$, if $u_{j,c} = 1$, then add (modulo 2) row r to row j. Set $t_r \leftarrow c$.
4. [loop] If $c \leq \ell$, then set $c \leftarrow c+1$ and go to Step 2.
5. [kernel] Evaluate the vector $\vec{K} = (\kappa_1, \ldots, \kappa_{\ell+1})$ defined by

$$\kappa_i = \begin{cases} u_{j,c} & \text{if } t_j = i > 0 \\ 1 & \text{if } i = c \\ 0 & \text{otherwise} \end{cases} .$$

   If $\kappa_{\ell+1} = 0$ and $c \leq \ell$, then set $c \leftarrow c+1$ and go to Step 2.
6. [output] If $\kappa_{\ell+1} = 1$, then output $\beta_i \leftarrow \kappa_i$ for all $1 \leq i \leq \ell$; otherwise, output `no solution`.

Example 1 (cont'd). If we apply the previous algorithm to Example 1, matrix $\mathbf{U}$ is given by

$$\mathbf{U} = \begin{pmatrix} \boxed{1} & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} .$$

We then successively obtain for $c = 1, 2, 3$

$$\mathbf{U} = \begin{pmatrix} \boxed{1} & 0 & 0 & 1 & 0 \\ 0 & \boxed{1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} \boxed{1} & 0 & 0 & 1 & 0 \\ 0 & \boxed{1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{1} & 0 & 1 \end{pmatrix}, \begin{pmatrix} \boxed{1} & 0 & 0 & 1 & 0 \\ 0 & \boxed{1} & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{1} & 0 & 1 \end{pmatrix}$$

after the elimination step. We also have $t_1 = 1$, $t_2 = 2$, $t_5 = 3$ (which is indicated by the boxes ($t_r = c$)) and $t_3 = t_4 = 0$. At this point, we have $c = 4$ and the kernel step yields the vector $\vec{K} = (u_{1,4}, u_{2,4}, u_{5,4}, 1, 0) = (1, 0, 0, 1, 0)$. Since $\kappa_5 = 0$, we increment c, $c = 5$, and go to the scanning step. Then, since $t_2 = t_5 = 0$ (note that $u_{c,2} = u_{c,5} = 1$), we directly go to the kernel step and obtain the new vector $\vec{K} = (u_{1,5}, u_{2,5}, u_{5,5}, 0, 1) = (0, 1, 1, 0, 1)$. So, we finally find $\beta_1 = 0$, $\beta_2 = 1$, $\beta_3 = 1$ and $\beta_4 = 0$. ♢

3.3 Improvements

The methods we have presented so far are subject to numerous possible improvements. We just mention two of these.

In Eq. (1), we consider only "messages" $\widehat{m}_i$ whose largest prime factor (modulo n) is p_B. As for modern factorization methods, a substantial speed-up can be obtained by also considering the $\widehat{m}_i$'s which are p_B-smooth except for one or two factors [15]. Another speed-up can be obtained by using structured Gaussian elimination to solve Eq. (11); see [18] for an efficient variation directly applicable to our case.

4 Generalizations

4.1 Higher exponents

The signature scheme presented in Section 2 can be generalized to other even public exponents besides $e = 2$. Define $\Lambda = \mathrm{lcm}[(p-1)/2, (q-1)/2]$. It suffices to choose e relatively prime to Λ, the corresponding private exponent d is then given according to $ed \equiv 1 \pmod{\Lambda}$ (see [24]). The scheme remains exactly the same except that $m' = s^2 \bmod n$ must be replaced by $m' = s^e \bmod n$ in the verification stage.

In that setting, Proposition 1 becomes

Proposition 2. *Let $n = pq$, where p, q are distinct primes and $p, q \equiv 3 \pmod 4$ and let e, d such that e is even,* $\gcd(e, \Lambda) = 1$ *and* $ed \equiv 1 \pmod{\Lambda}$. *If $(z|n) = -1$, then*

$$\gcd\bigl(z^{ed} \mp z \pmod n, n\bigr) = p \text{ or } q \ .$$

Proof. From $ed \equiv 1 \pmod{\Lambda}$, we deduce that $ed \equiv 1 \pmod{(p-1)/2}$ and so there exists $\gamma \in \mathbb{Z}$ such that $ed = \gamma \frac{p-1}{2} + 1$. Further, since $p \equiv 3 \pmod 4$, $(p-1)/2$ is odd. Hence, γ must be odd since ed is even. Consequently, $z^{ed} \equiv z^{\gamma(p-1)/2}\, z \equiv (z|p)^{\gamma}\, z \equiv (z|p)\, z \equiv \pm z \pmod p$ and similarly $z^{ed} \equiv (z|q)\, z \equiv -(z|p)\, z \equiv \mp z \pmod q$, which completes the proof. □

Since e is even, we can write $e = 2e_1$. It is here worth remarking that anyone can raise an element to the ${e_1}^{\text{th}}$ power (modulo n). So, if an adversary follows Steps (I)–(VI) as described in Section 3, she can recover the factors of n by computing

$$\gcd(S_\tau - \widehat{r}_\tau \pmod n, n)\,, \tag{12}$$

where

$$\begin{aligned} S_\tau := {s_\tau}^{e_1} \bmod n &\equiv ({\widehat{m}_\tau}^{d})^{e_1} \equiv \prod_{1 \le i \le \ell} \prod_{1 \le j \le B} {p_j}^{(\beta_i v_{j,i} - 4\gamma_j) d e_1} \\ &\equiv \prod_{1 \le i \le \ell} {s_i}^{\beta_i\, e_1} \prod_{1 \le j \le B} {p_j}^{-2\gamma_j} \pmod n \ . \end{aligned}$$

The forgery is thus no more expensive against a scheme with a large public exponent e than against the basic scheme with $e = 2$.

4.2 Higher degree schemes

Rabin-type schemes can be developed in any unique factorization domain. In [27], Williams presents a scheme, called M^3, with public exponent 3 using arithmetic in a quadratic number field. This scheme was later extended to cyclotomic fields by Scheidler and Williams in [21] where they also give a scheme with a public exponent 5. In [20], Scheidler modifies Williams' M^3 scheme so that it works with a larger class of primes. Finally, in [16], Loxton et al. give another cubic scheme; the main difference with [27] being its easy geometrical interpretation. In this paragraph, we will stick on this latter scheme because it most resembles the Rabin-Williams scheme presented in Section 2.

The scheme of Loxton et al. uses the ring of Eisenstein integers, namely $\mathbb{Z}[\omega]$, where $\omega = (-1+\sqrt{-3})/2$ is a primitive cube root of unity. Its correctness relies of the following lemma (compare it with Lemma 1).

Lemma 2. *Let $n = pq$, where p, q are distinct primes in $\mathbb{Z}[\omega]$, $3 \nmid \mathrm{N}n$ and $\mathrm{N}q \equiv 2\,\mathrm{N}p - 1 \pmod 9$. If $(z|n)_3 = 1$, then*

$$z^{(\mathrm{N}p-1)(\mathrm{N}q-1)/9} \equiv \left(\frac{z}{p}\right)_3^{(2/3)(\mathrm{N}p-1)} \pmod n \ .$$

Proof. See [16, Lemma 1]. □

Essentially, that scheme suggests to generate two primes $p, q \in \mathbb{Z}[\omega]$ such that $p \equiv 8 + 6\omega \pmod 9$ and $q \equiv 5 + 6\omega \pmod 9$ and to compute $n = pq$. The public key is n and the private key is $d = [(\mathrm{N}p-1)(\mathrm{N}q-1)+9]/27$, where N denotes the norm. A message m is signed by computing $\widetilde{m} = R(m)$ (for an appropriate function R) and $\widehat{m} = (1-\omega)^{3-t}\,[(1-\omega)\widetilde{m}+1]$, where $\omega^t = (\widetilde{m}|n)_3$; the signature is $s = \widehat{m}^d \bmod n$. The signature is then verified by cubing s, and if accepted, the message m is recovered.

We need an analogue of Proposition 1.

Proposition 3. *Let $n = pq$, where p, q are primes in $\mathbb{Z}[\omega]$, $\mathrm{N}p \equiv 7 \pmod 9$ and $\mathrm{N}q \equiv 4 \pmod 9$, and let $d = [(\mathrm{N}p-1)(\mathrm{N}q-1)+9]/27$. If $(z|n)_3 = \omega$ or ω^2, then*

$$\gcd\bigl(z^{3d} - \omega^k z \pmod n, n\bigr) = p \text{ or } q\,,$$

for some $k \in \{0,1,2\}$.

Proof. We first note that $(\mathrm{N}p-1)/3 \equiv 2 \pmod 3$ and $(\mathrm{N}q-1)/3 \equiv 1 \pmod 3$. So, $z^{3d} \equiv z^{(\mathrm{N}p-1)(\mathrm{N}q-1)/9}\,z \equiv (z|p)_3^{(\mathrm{N}q-1)/3}\,z \equiv (z|p)_3\,z \pmod p$, and similarly $z^{3d} \equiv (z|q)_3^{(\mathrm{N}p-1)/3}\,z \equiv (z|q)_3^{\,2}\,z \pmod q$. The proposition now follows by observing that $(z|p)_3 \neq (z|q)_3^{\,2}$ because, by hypothesis, $(z|n)_3 = (z|p)_3\,(z|q)_3 = \omega$ or ω^2. □

From this proposition, we can mimic the forgery presented against the Rabin-Williams scheme. The adversary has to find $\beta_1, \ldots, \beta_\ell$ such that (a) for $1 \le j \le$

B, $\sum_{1\le i\le \ell} \beta_i\, v_{j,i} \equiv 0 \pmod 3$; and (b) $\sum_{1\le i\le \ell} \beta_i\, \zeta_i \equiv 1,2 \pmod 3$, where $3\zeta_i := (\sum_{i\in\mathcal{J}(n,B)} v_{j,i}) \bmod 9$ and $\mathcal{J}(n,B) := \{1 \le j \le B : p_j \text{ is prime in } \mathbb{Z}[\omega] \text{ and } (p_j|n)_3 = \omega \text{ or } \omega^2\}$. She then computes $\vec{V}_\tau = (v_{1,\tau}, \ldots, v_{B,\tau}) \in (\mathbb{Z}/9\mathbb{Z})^B$, where $v_{j,\tau} = \sum_{1\le i\le \ell} \beta_i\, v_{j,i} - 9\gamma_j$ (for some $\gamma_j \in \mathbb{Z}$), $\widehat{r}_\tau = \prod_{1\le j\le B} {p_j}^{v_{j,\tau}/3} \pmod n$ and $s_\tau = \prod_{1\le i\le \ell} {s_i}^{\beta_i} \prod_{1\le j\le B} {p_j}^{-3\gamma_j} \bmod n$. Finally, by computing

$$\gcd(s_\tau - \omega^k\, \widehat{r}_\tau \pmod n, n) \tag{13}$$

for some $k \in \{0,1,2\}$, she finds the factors of n.

5 Applications

As an application, we will analyze the consequences of the previously described forgeries (Section 3) when the Rabin-Williams signature scheme is employed with the PKCS #1 v2.0 message encoding method as specified in [4]. We note, however, that the PKCS #1 standard is not expressly intended for use with the Rabin-Williams scheme but rather with the plain RSA scheme.

In the second part, we will present an algorithm which reduces the complexity of the forgery from n to $\sqrt{n}$. This algorithm is not restricted to PKCS #1: it remains applicable *whatever the employed encoding message method*.

5.1 PKCS #1 encoding method

We only briefly review the PKCS #1 message encoding method and refer the reader to [4] for details. PKCS #1 supports signature schemes with appendix. (In such a scheme, the message must accompany the signature in order to verify the validity of the signature.) The set of message representatives is thus given by $\mathcal{M}_R = \{\widetilde{m} \in (\mathbb{Z}/n\mathbb{Z})^* : \widetilde{m} \equiv 10 \pmod{16}\}$ (cf. Remark 1).

Let m be the message being encoded into the message representative $\widetilde{m}$. First, a hash function is applied to m to produce the hash value $H = \mathrm{Hash}(m)$. Next, the hash algorithm identifier and the hash value H are combined into an ASN.1 value and DER-encoded (see [11] for the relevant definitions). Let T denote the resulting DER-encoding. Similarly to what is done in ISO 9796 [3], we concatenate the octet $\mathtt{0A}_{16}$ to obtain the data string $D = T \| \mathtt{0A}_{16}$ (the reason is to ensure that D, viewed as an integer, is congruent to 10 modulo 16). The encoded message EM is then formed by concatenating the block-type octet BT, the padding string PS, the $\mathtt{00}_{16}$ octet and the data string D; or schematically,

$$EM = \mathtt{00}_{16} \| BT \| PS \| \mathtt{00}_{16} \| D\,, \tag{14}$$

where BT is a single octet containing the value $\mathtt{00}_{16}$ or $\mathtt{01}_{16}$. Let $\|n\|$ and $\|D\|$ respectively denote the octet-length of modulus n and D. When $BT = \mathtt{00}_{16}$ then PS consists of $\|n\| - \|D\| - 3$ octets having value $\mathtt{00}_{16}$; when $BT = \mathtt{01}_{16}$ then PS consists of $\|n\| - \|D\| - 3$ octets having value $\mathtt{FF}_{16}$.

The message representative $\widetilde{m}$ is the integer representation of EM.

Remark 4. Three hash functions are recommended: SHA-1 [1], MD2 [5], and MD5 [6]. The default function is SHA-1; MD2 and MD5 are only recommended for compatibility with existing applications based on PKCS #1 v1.5.

In what follows, we will assume that SHA-1 is the used hash function. In that case, the data string D consists of $(15 + 20 + 1) = 36$ octets ($= 288$ bits).

Type 0 encoding With type 0, i.e., when $BT = \mathtt{00}_{16}$, the message representatives, $\widetilde{m}_i$, are 288-bit long, *whatever the size of the modulus*. We have seen that the effectiveness of our forgeries is related to the smoothness (modulo n) of the $\widehat{m}_i$'s that appear in Eq. (1), where $\widehat{m}_i = \widetilde{m}_i$ or $\widetilde{m}_i/2$ according to the value of $(\widetilde{m}_i|n)$. So, when $BT = \mathtt{00}_{16}$, each $\widehat{m}_i$ is at most a 288-bit integer and we can hope that they factor (in $\mathbb{Z}$) into small primes.

Type 1 encoding Type 1 encoding, i.e., $BT = \mathtt{01}_{16}$, is the recommended way to encode a message. In that case, the message representatives are longer. For a 1024-bit modulus, these are 1016-bit integers (the leading octet in EM is $\mathtt{00}_{16}$). Therefore, the $\widehat{m}_i$'s happen unlikely to be smooth. But, what we ultimately need is to find smooth $\widehat{m}_i$'s *modulo* n, that is, we need to find an $\widehat{M}_i \equiv \widehat{m}_i \pmod{n}$ so that $\widehat{M}_i$ is smooth as an element in $\mathbb{Z}/n\mathbb{Z}$ (cf. Footnote ([1])). As already noted in [8], if the 1024-bit modulus has the special form

$$n = 2^{1023} \pm t\,, \tag{15}$$

then it is easy to find an $\widehat{M}_i$ whose magnitude is comparable to that of t. Indeed, when $BT = \mathtt{01}_{16}$, the padding string is formed with $(128 - 36 - 3) = 89$ octets having value $\mathtt{FF}_{16}$. Therefore, considering the data string D_i as a 288-bit integer, we can write from Eq. (14), $\widetilde{m}_i = \{(2^8)^{89} + [(2^8)^{89} - 1]\}(2^8)^{37} + D_i$. Hence, $\widehat{m}_i = (2^{713} - 1)2^{296} + D_i$ or $(2^{713} - 1)2^{295} + D_i/2$ according to $(\widetilde{m}_i|n) = 1$ or -1. So, defining $M_i := 2^{g_i}\,\widehat{m}_i - n$ and setting $\widehat{M}_i \equiv 2^{-g_i}\,M_i \equiv \widehat{m}_i \pmod{n}$, an appropriate choice for g_i will remove the term $2^{713+296}$ (resp. $2^{713+295}$). Namely, we set

$$M_i = \begin{cases} 2^{14}\,\widehat{m}_i - n = 2^{14}\,D_i - 2^{310} \mp t & \text{if } (\widetilde{m}_i|n) = 1 \\ 2^{15}\,\widehat{m}_i - n = 2^{15}\,D_i - 2^{310} \mp t & \text{if } (\widetilde{m}_i|n) = -1 \end{cases}\,. \tag{16}$$

Hence, since a special modulus of the form (15) with a square-free t as small as 400 bits offers the same security as a regular 1024-bit modulus [14], the M_i's as given in Eq. (16) can already be as small as 400-bit integers. Consequently, hoping that the M_i's factor into small integers (in $\mathbb{Z}$),[3] $\widehat{M}_i \equiv 2^{-g_i}\,M_i \equiv \widehat{m}_i \pmod{n}$ will be smooth as elements of $\mathbb{Z}/n\mathbb{Z}$.

[3] The probability that a 400-bit integer is smooth is relatively small; but see § 3.3 for some possible alternatives.

5.2 Arbitrary encoding

For general ("form-free") message encoding methods and moduli, the message representatives have roughly the same length as the modulus. We now present a generic method which on input an arbitrary message representative outputs a "message equivalent" whose length has the size of $\sqrt{n}$.

As in [8], we observe that if we can find integers a_i and b_i so that a_i is smooth and

$$M_i := a_i\,\widehat{m}_i - b_i\,n \tag{17}$$

is smooth as well, then $\widehat{M_i} :\equiv {a_i}^{-1}\,M_i \equiv \widehat{m}_i \pmod{n}$ is also smooth as an element of $\mathbb{Z}/n\mathbb{Z}$.
Explicitly, let $a_i = (-1)^{u_{0,i}} \prod_{1\le j\le B} {p_j}^{u_{j,i}}$ and $M_i = (-1)^{w_{0,i}} \prod_{1\le j\le B} {p_j}^{w_{j,i}}$ be the prime factorizations of a_i and M_i; then, we have

$$\widehat{M_i} \equiv \widehat{m}_i \equiv (-1)^{v_{0,i}} \prod_{1\le j\le B} {p_j}^{v_{j,i}} \pmod{n}\,, \tag{18}$$

where $v_{j,i} = u_{j,i} - w_{j,i}$, as required.

A related problem has been addressed by de Jonge and Chaum in [9, § 3.1] (see also [10]): they describe a method to find *small* integers a_i and M_i satisfying Eq. (17). The "Pigeonhole Principle" (e.g., see [22, Chapter 30]) quantifies how small can be a_i and M_i.

Proposition 4 (Pigeonhole Principle). *Let two integers $n, \widehat{m} \in \mathbb{Z}$. For any positive integer $A < n$, there exist integers a^* and b^* such that*

$$0 < |a^*| \le A \quad \textit{and} \quad |a^*\widehat{m} - b^*n| < \lceil n/A \rceil\ .$$

Proof. Consider the $(A+1)$ "pigeons" given by the numbers $P_a := a\,\widehat{m} \bmod n$ $(0 \le a \le A)$, i.e., each pigeon is an integer between 0 and $(n-1)$. We now form the A pigeonholes given by the integer intervals

$$\mathcal{I}_a = \begin{cases} \big[(a-1)\lceil n/A\rceil, a\lceil n/A\rceil\big[& \text{for } 1 \le a \le (A-1) \\ \big[(A-1)\lceil n/A\rceil, n-1\big] & \text{for } a = A \end{cases}\ .$$

Since there are more pigeons than pigeonholes, two pigeons are sitting in the same hole. Suppose these are pigeons P_x and P_y and that they are in hole $\mathcal{I}_a$; in other words, $P_x, P_y \in \mathcal{I}_a \iff |P_x - P_y| < \lceil n/A\rceil \iff |(x-y)\widehat{m} \bmod n| < \lceil n/A\rceil$. Noting that $0 \le x, y \le A$ and $x \ne y$, we set $a^* = x - y$ and so $0 < |a^*| \le A$. Hence, $|a^*\widehat{m} \bmod n| < \lceil n/A\rceil$ and the lemma follows by setting $b^* = \lfloor a^*\widehat{m}/n \rfloor$. □

In particular, taking $A = \lceil \sqrt{n}\,\rceil$, there exist $a_i, b_i \in \mathbb{Z}$ such that $|a_i| \le \lceil \sqrt{n}\,\rceil$ and $|M_i| \le \big\lceil n/\lceil \sqrt{n}\,\rceil\big\rceil - 1 = \big\lfloor n/\lceil \sqrt{n}\,\rceil\big\rfloor \le \lfloor \sqrt{n} \rfloor$. This means that for a given 1024-bit modulus n and *any* $\widehat{m}_i$ (corresponding to the message representative $\widetilde{m})_i$), there are integers a_i and b_i such both a_i and $M_i = a_i\,\widehat{m}_i - b_i\,n$ are 512-bit integers, *whatever the message encoding method*. It remains to explain how to find a_i and b_i. A simple means is given by the extended Euclidean algorithm [12, Algorithm X, p. 325].

Algorithm 2. On inputs n and $\widehat{m}_i \in \mathbb{Z}/n\mathbb{Z}$, this algorithm computes a solution a_i, b_i, M_i satisfying Eq. (17) so that $|a_i|$ and $|M_i|$ are $\leq \lceil \sqrt{n} \rceil$. It makes use of a vector (u_1, u_2, u_3) such that $u_1 \widehat{m}_i - u_2 n = u_3$ always holds.

1. [initialization] Set $(u_1, u_2, u_3) \leftarrow (0, -1, n)$ and $(v_1, v_2, v_3) \leftarrow (1, 0, \widehat{m}_i)$.
2. [Euclid] Compute $Q \leftarrow \lfloor u_3/v_3 \rfloor$ and $(t_1, t_2, t_3) \leftarrow (u_1, u_2, u_3) - (v_1, v_2, v_3)Q$. Then set $(u_1, u_2, u_3) \leftarrow (v_1, v_2, v_3)$ and $(v_1, v_2, v_3) \leftarrow (t_1, t_2, t_3)$.
3. [loop] If $u_3 > \lceil \sqrt{n} \rceil$, then return to Step 2.
4. [output] Output $a_i \leftarrow u_1$, $b_i \leftarrow u_2$ and $M_i \leftarrow u_3$.

Example 2. Using the modulus of Example 1 (i.e., $n = 26882749$), suppose we are given a message m_i whose encoding is $\widetilde{m}_i = 26543210$ ($\equiv 10 \pmod{16}$). Since $(\widetilde{m}_i | n) = -1$, we have $\widehat{m}_i = \widetilde{m}_i/2 = 13271605 = 5 \cdot 79 \cdot 33599$. This $\widehat{m}_i$ is not smooth, we thus apply Algorithm 2 and obtain $M_i = a_i \widehat{m}_i + b_i n$ with $a_i = 1821$, $b_i = 899$, $M_i = 1354$. We have $a_i = 3 \cdot 607$ and $M_i = 2 \cdot 677$. Hence, $\widehat{m}_i \equiv 2 \cdot 3^{-1} \cdot 607^{-1} \cdot 677 \pmod{n}$.

u_1 (a_i)	u_2 (b_i)	u_3 (M_i)
1	0	13271605
−2	−1	339539
79	*39*	*29584*
−871	−430	14115
1821	*899*	*1354*
−19081	−9420	575
39983	19739	204
−99047	−48898	167
139030	68637	37
−655167	−323446	19
794197	392083	18
−1449364	−715529	1

Note that Algorithm 2 does not always give the best possible solution. Considering all the steps of the extended Euclidean algorithm (see above), the best solution for our example is given by $a_i = 79$ and $M_i = 29584 = 2^4 \cdot 43^2$ which yields $\widehat{m}_i \equiv 2^4 \cdot 43^2 \cdot 79^{-1} \pmod{n}$. ♢

6 Conclusion

This paper presented a specialized version of the Coron-Naccache-Stern signature forgery. It applies to *any* Rabin-type signature scheme and to *any* (even) public verification exponent. Furthermore, contrary to the case of RSA, the forgery is *universal*: it yields the value of the private key.

References

1. FIPS 180-1. Secure Hash Standard. Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce, April 1995.
2. IEEE Std 1363-2000. IEEE Standard Specifications for Public-Key Cryptography. IEEE Computer Society, August 29, 2000.
3. ISO/IEC 9796-1. Information technology – Security techniques – Digital signature scheme giving message recovery, 1991.
4. PKCS #1 v2.0. RSA cryptography standard. RSA Laboratories, October 1, 1998. Available at http://www.rsasecurity.com/rsalabs/pkcs/.
5. RFC 1319. The MD2 message digest algorithm. Internet Request for Comments 1321, Burt Kaliski, April 1992. Available at http://www.ietf.org/rfc/rfc1319.txt.
6. RFC 1321. The MD5 message digest algorithm. Internet Request for Comments 1321, Ronald Rivest, April 1992. Available at http://www.ietf.org/rfc/rfc1321.txt.
7. Henri Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.
8. Jean-Sébastien Coron, David Naccache, and Julien P. Stern. On RSA padding. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 1999.
9. Wiebren de Jonge and David Chaum. Attacks on some RSA signatures. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 18–27, 1986.
10. Marc Girault, Philippe Toffin, and Brigitte Vallée. Computation of approximate L-th root modulo n and application to cryptography. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 110–117, 1990.
11. Burton S. Kaliski Jr. A layman's guide to a subset of ASN.1, BER, and DER. RSA Laboratories Technical Note, RSA Laboratories, November 1993. Available at http://www.rsasecurity.com/rsalabs/pkcs/.
12. Donald E. Knuth. *The Art of Computer Programming, v. 2. Seminumerical Algorithms*. Addison-Wesley, 2nd edition, 1981.
13. Kaoru Kurosawa, Toshiya Itoh, and Masashi Takeuchi. Public key cryptosystem using a reciprocal number with the same intractability as factoring a large number. *Cryptologia*, 12(4):225–233, 1988.
14. Arjen K. Lenstra. Generating RSA moduli with a predetermined portion. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag, 1998.
15. Arjen K. Lenstra and Mark S. Manasse. Factoring with two large primes. *Mathematics of Computation*, 63:785–798, 1994.
16. J. H. Loxton, David S. Khoo, Gregory J. Bird, and Jennifer Seberry. A cubic RSA code equivalent to factorization. *Journal of Cryptology*, 5(2):139–150, 1992.
17. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
18. Peter L. Montgomery. A block Lanczos algorithm for finding dependencies over GF(2). In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 106–120, 1995.
19. Michael O. Rabin. Digitized signatures and public-key functions as intractable as factorization. Technical Report LCS/TR-212, M.I.T. Lab. for Computer Science, January 1979.

20. Renate Scheidler. A public-key cryptosystem using purely cubic fields. *Journal of Cryptology*, 11(2):109–124, 1998.
21. Renate Scheidler and Hugh C. Williams. A public-key cryptosystem utilizing cyclotomic fields. *Designs, Codes and Cryptography*, 6:117–131, 1995.
22. Joseph H. Silverman. *A Friendly Introduction to Number Theory.* Prentice-Hall, 1997.
23. Robert D. Silverman and David Naccache. Recent results on signature forgery, April 1999. Available at http://www.rsasecurity.com/rsalabs/bulletins/sigforge.html.
24. Hugh C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, IT-26(6):726–729, 1980.
25. ———. Some public-key crypto-functions as intractable as factorization. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 66–70. Springer-Verlag, 1986.
26. ———. Some public-key crypto-functions as intractable as factorization. *Cryptologia*, 9(3):223–237, 1985. An extended abstract appears in [25].
27. ———. An M^3 public key encryption scheme. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 358–368. Springer-Verlag, 1986.

On the Power of Misbehaving Adversaries and Security Analysis of the Original EPOC

[Published in D. Naccache, Ed., *Topics in Cryptology – CT-RSA 2001*, vol. 2020 of *Lecture Notes in Computer Science*, pp. 208–222, Springer-Verlag, 2001.]

Marc Joye[1], Jean-Jacques Quisquater[2], and Moti Yung[3]

[1] Gemplus Card International, Gémenos, France
marc.joye@gemplus.com
[2] UCL Crypto Group, Louvain-la-Neuve, Belgium
jjq@dice.ucl.ac.be
[3] CertCo, New York NY, U.S.A.
moti@certo.com, moti@cs.columbia.edu

Abstract. Nowadays, since modern cryptography deals with careful modeling and careful proofs, there may be two levels of cryptanalysis. One, the traditional breaking or weakness demonstration in schemes which are not provably secure. The second level of cryptanalysis, geared towards provably secure schemes, has to do with refining models and showing that a model was either insufficient or somewhat unclear and vague when used in proving systems secure. The best techniques to perform this second type of investigation are still traditional cryptanalysis followed by corrections. In this work, we put forth the second type of cryptanalysis.

We demonstrate that in some of the recent works modeling chosen ciphertext security (non-malleability), the notion of validity of ciphertext was left vague. It led to systems where under the model as defined/understood, it was shown provably secure. Yet, under another (natural) behavior of the adversary, the "provably secure system" is totally broken, since key recovery attack is made possible. We show that this behavior of an adversary is possible and further there are settings (the context of escrowed public key cryptosystems) where it is even highly relevant.

We mount the attack against systems which are chosen-ciphertext secure and non-malleable (assuming the adversary probes with valid messages), yet they are "universally" insecure against this attack: namely, the trapdoor key gets known by the adversary (as in Rabin's system under chosen ciphertext attacks). Specifically, the attack works against EPOC which has been considered for standardization by IEEE P1363 (the authors have already been informed of the attack and our fix to it and will consider this issue in future works). This re-emphasizes that when proving chosen-ciphertext security, allowing invalid ciphertext probes increases the adversary's power and should be considered as part of the model and in proofs.

1 Introduction

Classifying the security of cryptosystems, based on the power of the attacking adversary, is a central subject in modern cryptography. After many years of work by many researchers, the notion of attacks on public key systems has been carefully presented in a unified way in [2, 5]. In the attack modeling of chosen ciphertext attacks they only explicitly consider *valid ciphertexts* by the adversary, referring directly to the size of the ciphertexts used by the adversary. —In a later (final) versions they justify that: an adversary who sends "invalid ciphertexts" will know that the machine it probes will answer that the ciphertext is invalid as a justification for this model (this was published on the web, but since our results here were made known in Feb. 2000 (see [A7]), this was omitted, by now). In any case, the model (even in these careful elegant classification works) has left vague and has not directly treated how to deal with invalid ciphertext. Such vagueness is dangerous since at times it may lead to misinterpretations and potentially to false claims based on correct proofs (as we will show). Our purpose here is to demonstrate and thus to re-emphasize that it is, in fact, important to deal with invalid ciphertext probing by the adversary. We do this via cryptanalysis which employs such messages. Since our attack is against a scheme provably secure against attacker which only employs valid ciphertext, we demonstrate that this issue is not merely for completeness of modeling, but a central one which should be considered in proofs, when chosen-ciphertext attacks are allowed. In more general terms, the work demonstrates how important is the interaction between careful modeling and investigating (seemingly) extended settings and new scenarios in order to refine, better understand and eliminate vagueness in formal models.

Security Notions under Active Attacks. The notions of "chosen ciphertext security" [CCS] (in a non-adaptive [36] and an adaptive [43, 16] fashion) and "non-malleability" [NM] [16] are security notions for cryptosystems when coping with an active probing by an adversary who tries to break a system (namely, understand a message [CCS] or modify it [NM]). The adversary can choose ciphertexts in a certain way and probe the device on these messages. The security implies that the attacker does not get any advantage in breaking the system due to the probing. These security notions are extensions of "semantic security" (or polynomial security) [25] which assures that the system is secure —hiding all partial information against a passive adversary (in the public key model a passive adversary can, by itself, mount a chosen message attack).

The first public encryption scheme provably secure against (non-adaptive) chosen ciphertext attacks was devised by Naor and Yung [36] in 1990. In [43], Rackoff and Simon generalized their results and realized the first scheme provably secure against adaptive attacks. In the same year (1991), Dwork, Dolev and Naor [16] gave another provably secure scheme. More practical constructions (some of which are heuristics and some are validated in idealized random hash models) were proposed by Damgård [12] (only secure against non-adaptive

attacks [47]), Zheng and Seberry [47] (see also [3] and [33]), Lim and Lee [33] (cryptanalyzed in [19]), Bellare and Rogaway [3, 4] and Shoup and Gennaro [45] (for threshold cryptography). Recent formal treatment of the issue was given by Bellare, Desai, Pointcheval and Rogaway and Bellare and Sahai [2, 5]; they show, among other things that under adaptive chosen message attacks indistinguishability attack is equivalent to malleability one. Recently designed schemes which are practical and based on new assumption or hybrid encryption are given in [40, 23, 39, 41]. The security of these practical schemes holds in the idealized random oracle setting [3] and/or under non-standard assumptions. One notable exception is the Cramer-Shoup scheme [11] which remarkably achieves both provable security (under the decisional Diffie-Hellman assumption, namely in the standard model) and high level of practicality.

The Attack. We now define somewhat more formally our attack. Roughly speaking, it is a chosen ciphertext attack where the adversary has access to a "decryption oracle." It however emphasizes and explicitly allows the adversary to misbehave and repeatedly feed the decryption oracle with invalid ciphertexts. (Remark: we use "our attack", though, of course, we do not claim it is a new (see [6]), just that using it against provable systems and emphasizing it in contrast with the context which uses only valid messages are, as far as we know, new).

Definition 1 (The Attack). *Let k be a security parameter that generates matching encryption/decryption keys (e, d) for each user in the system. A* chosen-ciphertext attack *is a process which, on input 1^k and e, obtains*

- *either plaintexts (relatively to d) corresponding to ciphertexts of its choice; or*
- *an indication that the chosen ciphertexts are invalid,*

for polynomially (in k) many ciphertexts, and produces an history tape h.

To this attack corresponds a security notion, namely resistance against our attack which coincides with chosen ciphertext security. A probabilistic polynomial time machine, called "message finder", generates two messages m_1 and m_2 on input 1^k and an auxiliary tape (which may include h, e and other public information). Let c be the ciphertext corresponding to m_b where b is randomly drawn from $\{0, 1\}$. Then, given m_1, m_2, c, h and e, another probabilistic polynomial time algorithm, called "message distinguisher", outputs $b' \in \{0, 1\}$. The (non-adaptive) chosen ciphertext attack *succeeds* if $b = b'$. Similarly to [43], we can make the previous scenario stronger by assuming that the adversary may run a second chosen ciphertext attack upon receiving the challenge ciphertext c (the only restriction being that the adversary does not probe on c). Accordingly, this adaptive attack *succeeds* is $b = b'$.

We may even reduce the attacker's probing power by letting it know if the ciphertext corresponds to a valid message or not.

Definition 2 (Security). *An encryption scheme is* secure *if every (non-adaptive /adaptive) chosen ciphertext attack succeeds with probability at most negligibly greater than 1/2.*

Our Results. We first apply the attack model to break the EPOC systems [37, 38]. These are very interesting systems which are about three year old and which have a lot of insight behind them (i.e., they use new trapdoor). They are provably secure against adaptive chosen ciphertext attacks in the ideal hash model. So indeed, if the context is such that our adversary is excluded, these are high quality ciphers (they are under consideration for standardization in IEEE P1363a). Yet, we teach that there are extended situations (i.e., misbehaving adversaries) where more care is needed since the systems are broken in these cases. We then show that even interactive systems which are secure against traditional chosen ciphertext attacks, can fail against the extended setting. We then discuss measures for correcting the schemes in order to prevent the attacks (which demonstrates the importance of the original work on these schemes). Finally, we revisit the general implications of the attack on chosen ciphertext security. Finally, we comment that we have notified the authors of EPOC of the attacks and the vagueness of the definitions, and they took notice. The EPOC authors' reaction is presented in an Appendix.

An Application of the Model. How realistic is to allow explicit invalid ciphertext and how much one should care about these? One can argue that when attacking a server system to provide decryptions of ciphertexts, then if too many invalid ones are asked, the server may shuts itself up. This may lead to denial of service attacks. Even more so, the attack is always possible in the context of escrow public key systems (for the sake of law enforcement). See Section 4 for details.

2 The Attacks

The attack which can be called "chosen valid/invalid ciphertext attack" applies to a large variety of cryptosystems, including systems using the so-called "coset encryption" [42]. See [24] for an application to the 'RSA for paranoids' [44] and [29] for the NICE [27] and HJPT [28] systems.

The above are attacks on "raw algebraic versions" of trapdoor functions. Perhaps other purely algebraic trapdoors are susceptible to the attack. However, more interestingly and perhaps somewhat surprising, we actually illustrate in this section attacks on a public encryption system which already possesses very strong security properties. The scheme is the system by Okamoto, Uchiyama and Fujisaki, EPOC [38]. EPOC has two versions, EPOC-1 and EPOC-2, and uses the trapdoor function described in [37]. It presents the advantages of being secure and non-malleable under chosen-ciphertext attacks, which, following [2], represents the *highest* level of security. Moreover, we show that interactive protocols [17] aiming to transform a semantically secure system into a system secure against chosen-ciphertext attacks may also be susceptible to the attack.

2.1 The EPOC-1 System

Hereafter, we give a brief review of EPOC-1; we refer to [38] for details. The scheme is divided into three parts: system setup, encryption and decryption.

[System setup] For security parameter k, two k-bit primes p and q are chosen and $n = p^2q$. Then an element $g \in (\mathbf{Z}/n\mathbf{Z})^\times$ such that $g_p = g^{p-1} \bmod p^2$ has order p is chosen randomly. Likewise $h_0 \in (\mathbf{Z}/n\mathbf{Z})^\times$ is chosen randomly (and independently from g) and $h = (h_0)^n \bmod n$. Finally, three integers p_{Len}, m_{Len} and r_{Len} such that $p_{\text{Len}} = k$ and $m_{\text{Len}} + r_{\text{Len}} \leq p_{\text{Len}} - 1$ and a public (hash) function H are defined.
The public parameters are $(n, g, h, p_{\text{Len}}, m_{\text{Len}}, r_{\text{Len}}, H)$. The secret parameters are (p, g_p).

[Encryption] A message $M \in \{0,1\}^{m_{\text{Len}}}$ is encrypted as

$$C = g^{(M\|R)} h^r \bmod n$$

where R is uniformly chosen in $\{0,1\}^{r_{\text{Len}}}$ and $r = H(M\|R)$.

[Decryption] Given the ciphertext C, the decryption process runs as follows. Let

$$X = \frac{\mathrm{L}(C_p)}{\mathrm{L}(g_p)} \bmod p$$

where $C_p = C^{p-1} \bmod p^2$ and $\mathrm{L}(x) = (x-1)/p$. Then if $g^X h^{H(X)} \bmod n = C$ holds, the decrypted message is given by $[X]^{m_{\text{Len}}}$ (that is, the m_{Len} most significant bits of X); otherwise the null string ε is output.

2.2 The Attack

The encryption process assumes that the message being encrypted is smaller than $2^{m_{\text{Len}}}$, or more precisely that $(M\|R) < 2^{p_{\text{Len}}-1}$. What happens if a larger message is encrypted?

Let $\hat{C}$ $(= g^{(\hat{M}\|R)} h^{H(\hat{M}\|R)} \bmod n)$ denote the ciphertext corresponding to a message $\hat{M}$. The decryption of $\hat{C}$ yields the intermediary value

$$X = \frac{\mathrm{L}(\hat{C}^{p-1} \bmod p^2)}{\mathrm{L}(g_p)} \bmod p \ .$$

Defining $\hat{X} = (\hat{M}\|R)$, we have $X = \hat{X} \bmod p$; or equivalently $\hat{X} = X + \alpha p$ with $\alpha = \lfloor \hat{X}/p \rfloor$. If $\hat{X} \geq p$ then $\hat{X} \neq X$ (i.e., $\alpha > 0$) and the test $g^X h^{H(X)} \bmod n \stackrel{?}{=} \hat{C}$ will fail. The decryption algorithm will thus output the null string ε. This can be exploited by an adversary as follows. Since the secret prime p is a p_{Len}-bit number, she knows that p lies in the interval $I_0 =]2^{p_{\text{Len}}-1}, 2^{p_{\text{Len}}}[$. So, she chooses a message $\hat{M}$ such that $\hat{X} = (\hat{M}\|R) \in I_0$ and computes the corresponding ciphertext $\hat{C}$. If $\hat{C}$ can be decrypted then she knows that $\hat{X} < p$; otherwise (i.e., if ε is returned) she knows that $\hat{X} \geq p$. She then reiterates the process with the interval $I_1 =]\hat{X}, 2^{p_{\text{Len}}}[$ or $I_1 =]2^{p_{\text{Len}}-1}, \hat{X}]$, respectively. And so on... until

the interval becomes small enough to guess —by exhaustion or more elaborated techniques (e.g., [10, 8])— the correct value of p. Noting that each iteration of a standard binary search halves the interval containing p, an upper bound for the total number of probes is certainly $p_{\text{Len}} - 1$. For example, with a 1024-bit modulus n, at most 340 ciphertexts are necessary to recover the whole secret key.

2.3 The EPOC-2 System

In EPOC-2, the system setup is broadly the same as in EPOC-1 except that two public (hash) functions H and G are defined together with a symmetric cryptosystem. We let $\text{SymEnc}(K, X)$ (resp. $\text{SymDec}(K, X)$) denote the encryption (resp. decryption) of X under the symmetric key K. A message $M \in \{0,1\}^{m_{\text{Len}}}$ is encrypted as (C_1, C_2) with $C_1 = g^R h^{H(M\|R)} \bmod n$ and $C_2 = \text{SymEnc}(G(R), M)$ where R is uniformly chosen in $\{0,1\}^{r_{\text{Len}}}$. Given (C_1, C_2), the decryption algorithm computes $C_p = C_1^{p-1} \bmod p^2$, $R' = \frac{\text{L}(C_p)}{\text{L}(g_p)} \bmod p$ and $M' = \text{SymDec}(G(R'), C_2)$. If $g^{R'} h^{H(M'\|R')} \equiv C_1 \pmod n$ then the plaintext is M'; otherwise the null string ε is output. So, the attack on EPOC-1 readily applies on EPOC-2. The adversary now guesses the value of the secret factor p according to $p > R$ if the decryption process is possible or $p \leq R$ if ε is returned, from suitable values of R she chooses.

2.4 The Fischlin PPTK Protocol

In [17], R. Fischlin presents a generic technique to turn any semantically secure cryptosystem into an (interactive) scheme which is immune against chosen-ciphertext attacks. We will apply this technique to the (semantically secure) Okamoto-Uchiyama cryptosystem [37]. The resulting scheme is very similar to the EPOC-1 system. This is not too surprising if you know that the EPOC systems are derived from an application to the Okamoto-Uchiyama system of the generic techniques of [21] (see also [22]) to transform a semantically secure system into a system secure against chosen-ciphertext attacks.

[System setup] For security parameter k, the parameters p, q, n, g, g_p, h_0 and h are defined as in § 2.1. There are also two integers p_{Len} and m_{Len} such that $p_{\text{Len}} = k$ and $2m_{\text{Len}} \leq p_{\text{Len}} - 1$. The public parameters are $(n, g, h, p_{\text{Len}}, m_{\text{Len}})$. The secret parameters are (p, g_p).

[Commitment/Encryption] A sender commits to a message $M \in \{0,1\}^{m_{\text{Len}}}$ by computing and sending

$$C = g^{(M\|R)} h^r \bmod n$$

where R is uniformly chosen in $\{0,1\}^{m_{\text{Len}}}$ and r in $\{0,1\}^{2m_{\text{Len}}}$. Note that C is the Okamoto-Uchiyama encryption of $(M\|R)$.

[Challenge] Upon receiving C, the receiver chooses a challenge $\lfloor p_{\text{Len}}/2 \rfloor$-bit prime π which he sends to the sender.

[PPTK] The sender computes $X_\pi = (M\|R) \bmod \pi$ and sends it to the receiver as a proof of plaintext knowledge.

[Decryption] Given X_π, the receiver decrypts C as

$$X = \frac{\mathrm{L}(C_p)}{\mathrm{L}(g_p)} \bmod p$$

where $C_p = C^{p-1} \bmod p^2$. Then if $X \equiv X_\pi \pmod{\pi}$, he accepts the plaintext given by $[X]^{m_{\mathrm{Len}}}$ (that is, the m_{Len} most significant bits of X); otherwise the null string ε is output, i.e., the receiver rejects the encryption.

The idea behind Fischlin's technique is quite intuitive. To make a system immune against chosen-ciphertext attacks, the sender (interactively) provides a "proof of plaintext knowledge" (PPTK). Although this seems sound, the attack presented against EPOC-1 in § 2.2 still applies. If $(M\|R)$ is smaller than the secret prime p then the decryption of the commitment C, X, is equal to $(M\|R)$. Therefore, the relation $X \equiv (M\|R) \pmod{\pi}$ will be verified whatever the value of the challenge π is. On the contrary, if $(M\|R) \geq p$ then the verification will fail and the null string ε is returned. So as before, the adversary can recover the bits of p successively according to whether ε is returned or not from appropriately chosen values for M. (Remark: recently, the author has removed his paper [17] from the public library, yet we do not think that it is due to the attack since the scheme as a generic method may be sound once considering the issues raised in the current work and similar considerations, see our repair to the specific application below.)

3 Repairing the Schemes

Here we show how to repair the systems, thus showing the usefulness of the work on the original schemes (the standardization bodies have to take note of our fixes, though).

The attack, as presented in § 2.2, is easily avoidable. EPOC-1 requires that message M being encrypted is such that $X = (M\|R) < 2^{p_{\mathrm{Len}}-1}$. This condition can be explicitly checked at the decryption stage:

[Decryption] Given the ciphertext C, the decryption process runs as follows. Let

$$X = \frac{\mathrm{L}(C_p)}{\mathrm{L}(g_p)} \bmod p$$

where $C_p = C^{p-1} \bmod p^2$ and $\mathrm{L}(x) = (x-1)/p$. Then if $g^X h^{H(X)} \bmod n = C$ **and if $X < 2^{p_{\mathrm{Len}}-1}$ holds**, the decrypted message is given by $[X]^{m_{\mathrm{Len}}}$ (that is, the m_{Len} most significant bits of X); otherwise the null string ε is output.

Now the attacker has no longer advantage to feed the decryption oracle with invalid ciphertexts $\hat{C}$ (i.e., corresponding to an $\hat{X} \geq 2^{p_{\mathrm{Len}}-1}$). Indeed, if $\hat{X} \in$

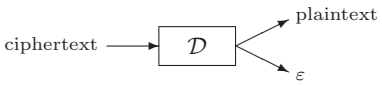


Fig. 1. Decryption algorithm.

$[2^{p_{\text{Len}}-1}, p[$ then the decryption process yields an $X = \hat{X} \geq 2^{p_{\text{Len}}-1}$ and so the null string ε is returned. If $\hat{X} \geq p$ then $X \neq \hat{X}$ (and thus $g^X h^{H(X)} \bmod n \neq \hat{C}$) and again ε is returned.

Likewise, EPOC-2 can be made robust against the attack of § 2.3 by further checking that $R' < 2^{r_{\text{Len}}}$ $(< 2^{p_{\text{Len}}-1}$) in the decryption stage. Finally, in Fischlin protocol, the receiver must also check that $X < 2^{p_{\text{Len}}-1}$ in the decryption stage and reject the encryption if it is not the case.

4 Illustration: The "Policeman-in-the-middle Attack"

In this section, we present a detailed example in the context of escrowed public key cryptosystems. The attack is by misbehaving law enforcement which fakes ciphertexts repeatedly, and asks the escrow authorities to recover them (thus the proposed name of the attack: "the Policeman-in-the-middle Attack"). The attacker is allowed to misbehave and choose "invalid ciphertexts" (since, supposedly, they are what the wiretapping has recorded and this fact has to be reported).

The basic configuration of the system model (when concentrating on a single sender-receiver pair) is given in Fig. 2. It includes a sender (Alice) which employs the receiver's (Bob) public key to send messages. The receiver gets the ciphertext message and can decrypt it. In addition, the law enforcement (Police) gets the message and forwards it to the escrow agent (TTP). Police gets back a cleartext which is the valid decryption of the message or an indication of "invalid message" from TTP. (Typically, Police is authorized to decrypt messages in some time interval and based on this authorization by the court, TTP has to comply and serve as a "decryption oracle" say at some time interval.) The weaker probing capability where the trusted party only answers whether a ciphertext correspond to a valid or invalid message (which suffices for our attacks), is realistic in the context in which sporadic tests of compliance with the escrow system are performed by law enforcement and the TTP only validates correct usage.

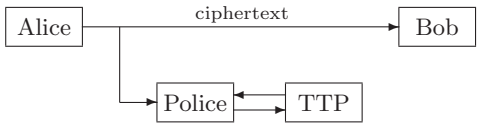


Fig. 2. Basic model.

Related Work on Escrow Systems.

Indeed, the notion of the attack makes sense in the context of the Police which tries to verify messages and the sender and the receiver may be bypassing the system. Therefore, the knowledge of "invalid message" is important (and should be supplied) to law enforcement. This is an interesting interplay between a protocol notion (escrowed encryption) and the relevant attack (chosen valid/invalid ciphertext attack). Let us review (only) some of the various escrow systems models which have been considered in the literature. A quite general framework to describe key escrow systems was proposed in [13] by Denning and Brandstad. Upon this, they classified the escrow mechanisms of complete systems as well as various design options, including the Escrow Encryption Standard (EES) and its Clipper implementation [1, 14] (see also [7, 35]), the fair cryptosystems [34, 31], the traceable ciphertexts [15, 32, 9], the Trusted Third Parties services [30], etc... (See also [18] for further discussions.) The model of Denning and Brandstad assumes that the sender (Alice) binds the ciphertext and the corresponding encryption key, normally by attaching a "data recovery field" (DRF) to the ciphertext. In our model, the DRF is merely an indication that the ciphertext was encrypted under Bob's public key. Variants on this model were considered in [20] by Frankel and Yung. They abstracted a public key based model where a message is sent to two receivers and where validation is added so that the line contains messages that have been validated as "messages available to both Bob and Police", then such systems are equivalent to "chosen ciphertext secure public-key systems," and furthermore, the reductions are very efficient (security wise).

5 Chosen Valid/Invalid Ciphertext Attacks

The scheme of Damgård [12] is semantically secure and has some other heuristic security properties, but a man-in-the-middle attack shows that this scheme is malleable [46, § 6]. EPOC is semantically secure and was "shown" to be non-malleable but is susceptible to a policeman-in-the-middle attack. This emphasizes the extended notion of chosen ciphertext security which considers security under "chosen **valid/invalid** ciphertext attacks." Certain security proofs assume that the adversary gets no credit for producing an invalid ciphertext. While this is true for most cryptosystems indeed, this is incorrect in general.

A particularity of Okamoto-Uchiyama primitive (as well as the other coset-based encryption primitives) is that the whole set of valid messages, $[0, p)$, is kept secret. Thus, to construct a cryptosystem thereof, one must work in a subset $[0, T)$ with $T < p$. This gives rise to two kinds of invalid ciphertexts: the invalid ciphertexts (i.e., those for which the null string ε is returned) and those for which a message is returned rather than a notification of invalidity. This shows the soundness of our repair (Section 3) since ε is returned for both types of invalid ciphertexts.

In many of the "generic constructions" there is a polynomial time algorithm so that when given a ciphertext it can verify (with overwhelming probability)

that we have a "proper ciphertext" which implies that it is a valid plaintext which is encrypted correctly (e.g., the constructions that employ general non-interactive zero-knowledge as in [36, 43]). Thus implicitly, either one sends valid ciphertext or the ciphertext can be rejected in polynomial-time (namely, without the computational power of the decryption algorithm). In this case indeed "invalid ciphertexts" do not add power (the probing adversary can reject the invalid ciphertext itself). However, as demonstrated here this may not be the case with other schemes where there is no public verification of ciphertext validity.

Sometimes, considering only valid messages may be enough. For example, for the concrete schemes we attack (EPOC), it may still be very useful in cases where the tampering adversary attacks a centralized device (the device may stop on the first invalid message, or may record and limit such attacks). In this setting the security as was proved in [38] applies. However, in the protocol setting we identified, reporting "invalid ciphertext" is part of the actual task of the decryption entity (escrow authorities or TTP). We conclude that in these cases the systems have to be robust against the extended setting.

Acknowledgements

The authors are grateful to Jean-Sébastien Coron and David Pointcheval for some fruitful comments.

References

1. FIPS PUB 185. Escrowed encryption standard (EES). Federal Information Processing Standards Publication 185, U.S. Dept of Commerce, February 1994.
2. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. Full paper (30 pages), available at URL `<http://www-cse.ucsd.edu/users/mihir/papers/pke.html>`, February 1999. An extended abstract appears in H. Krawczyk, ed., *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, Springer-Verlag, 1998.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, ACM Press, 1993.
4. ———. Optimal asymmetric encryption. In A. De Santis, ed., *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Springer-Verlag, 1995.
5. Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions. In M. Wiener, ed., *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536, Springer-Verlag, 1999.
6. Daniel Bleichenbacher. A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS#1. In H. Krawczyk, ed., *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, Springer-Verlag, 1998.
7. Matt Blaze. Protocol failure in the escrowed encryption standard. In *Proc. of the 2nd ACM Conference on Computer and Communications Security*, pages 59–67, ACM Press, 1994.

8. Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. Factoring $N = p^r q$ for large r. In J. Stern, ed., *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, Springer-Verlag, 1999.
9. Colin Boyd. Enforcing traceability in software. In Y. Han, T. Okamoto, and S. Qing, eds., *Information and Communications Security (ICICS '97)*, volume 1334 of *Lecture Notes in Computer Science*, pages 398–408, Springer-Verlag, 1997.
10. Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In U. Maurer, ed., *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 178–189, Springer-Verlag, 1996.
11. Ronald Cramer and Victor Shoup. A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, ed., *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Springer-Verlag, 1998.
12. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, ed., *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456, Springer-Verlag, 1992.
13. Dorothy E. Denning and Dennis K. Brandstad. A taxonomy for key escrow encryption systems. *Communications of the ACM*, 39(3):34–40, 1996.
14. Dorothy E. Denning and Miles Smid. Key escrowing today. *IEEE Communications Magazine*, 32(9):58–68, 1994.
15. Yvo Desmedt. Securing traceability of ciphertexts: Towards a secure software key escrow system. In L. C. Guillou and J.-J. Quisquater, eds., *Advances in Cryptology – EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 147–157, Springer-Verlag, 1995.
16. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. Manuscript (51 pages), available at URL `<http://www.wisdom.weizmann.ac.il/~naor/onpub.html>`, To appear in *SIAM J. on Computing*. A preliminary version appears in *Proc. of the 23rd ACM Annual Symposium on the Theory of Computing (STOC '91)*, pages 542–552, ACM Press, 1991.
17. Roger Fischlin. Fast proof of plaintext-knowledge and deniable authentication based on Chinese remainder theorem. Theory of Cryptography Library, 99-06, available at URL `<http://philby.ucsd.edu/cryptolib/1999.html>`, March 1999.
18. Yair Frankel and Moti Yung. Escrow encryption systems visited: attacks, analysis and designs. In D. Coppersmith, ed., *Advances in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 222–235, Springer-Verlag, 1995.
19. ———. Cryptanalysis of the immunized LL public key systems. In D. Coppersmith, ed., *Advances in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 287–296, Springer-Verlag, 1995.
20. ———. On characterization of escrow encryption schemes. In P. Degano, R. Garriero, and A. Marchetti-Spaccamela, eds., *Automata, Languages, and Programming (ICALP '97)*, volume 1256 of *Lecture Notes in Computer Science*, pages 705–715, Springer-Verlag, 1997.
21. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In H. Imai and Y. Zheng, eds., *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68, Springer-Verlag, 1999.
22. ———. How to enhance the security of public-key encryption at minimum cost. *IEICE Transactions on Fundamentals*, E83-A(1): 24–32, 2000.

23. _____. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, ed., *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Springer-Verlag, 1999.
24. Henri Gilbert, Dipankar Gupta, Andrew Odlyzko, and Jean-Jacques Quisquater. Attacks on Shamir's 'RSA for paranoids'. *Information Processing Letters*, 68: 197–199, 1998.
25. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270-299, 1984.
26. Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction attacks against several public-key cryptosystems. In V. Varadharajan and Yi Mu, eds., *Information and Communications Security*, volume 1726 of *Lecture Notes in Computer Science*, pages 2–12, Springer-Verlag, 1999.
27. Michael Hartmann, Sachar Paulus, and Tsuyoshi Takagi. NICE - New ideal coset encryption. In Ç. K. Koç and C. Paar, eds., *Cryptographic Hardware and Embedded Systems*, volume 1717 of *Lecture Notes in Computer Science*, pages 328–339, Springer-Verlag, 1999.
28. Detlef Hühnlein, Michael J. Jacobson Jr., Sachar Paulus, and Tsuyoshi Takagi. A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption. In K. Nyberg, ed., *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 294–307. Springer-Verlag, 1998.
29. Éliane Jaulmes and Antoine Joux. A NICE cryptanalysis. In B. Preneel, ed., *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 382–391, Springer-Verlag, 2000.
30. Nigel Jefferies, Chris Mitchell, and Michael Walker. A proposed architecture for trusted third parties services. In E. Dawson and J. Golić, eds., *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 98–104, Springer-Verlag, 1996.
31. Joe Kilian and Tom Leighton. Fair cryptosystems, revisited. In D. Coppersmith, ed., *Advances in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 208–221, Springer-Verlag, 1995.
32. Lars R. Knudsen and Torben P. Pedersen. On the difficulty of software key escrow. In U. Maurer, ed., *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 237–244, Springer-Verlag, 1996.
33. Chae Hoon Lim and Pil Joong Lee. Another method for attaining security against chosen ciphertext attacks. In D. R. Stinson, ed., *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 420–434, Springer-Verlag, 1994.
34. Silvio Micali. Fair public-key cryptosystems. In E. F. Brickell, ed., *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 113–138, Springer-Verlag, 1993.
35. Silvio Micali and Ray Sidney. A simple method for generating and sharing pseudo-random functions, with applications to Clipper-like key escrow systems. In D. Coppersmith, ed., *Advances in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 185–196, Springer-Verlag, 1995.
36. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proc. of the 22nd ACM Annual Symposium on the Theory of Computing (STOC '90)*, pages 427–437, ACM Press, 1990.
37. Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, ed., *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318. Springer-Verlag, 1998.

38. Tatsuaki Okamoto, Shigenori Uchiyama, and Eiichiro Fujisaki. EPOC: Efficient probabilistic public-key encryption. Submission to P1363a, available at URL `<http://grouper.ieee.org/groups/1363/addendum.html>`, November 1998.
39. Pascal Paillier and David Pointcheval. Efficient public-key cryptosystem provably secure against active adversaries. In *Advances in Cryptology – ASIACRYPT '99*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, 1999.
40. David Pointcheval. New public-key cryptosystem based on the dependent-RSA problem. In J. Stern, ed., *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, Springer-Verlag, 1999.
41. ———. Chosen ciphertext security for any one-way cryptosystem. In H. Imai and Y. Zheng, eds., *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 129–146, Springer-Verlag, 2000.
42. Sachar Paulus and Tsuyoshi Takagi. A generalization of the Diffie-Hellman problem and related cryptosystems allowing fast decryption. In *Proc. of the 1998 International Conference on Information Security and Cryptology (ICISC '98)*, Seoul, December 18–19, 1998.
43. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, ed., *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, Springer-Verlag, 1992.
44. Adi Shamir. RSA for paranoids. *Cryptobytes*, 1(2):1–4, 1995.
45. Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. IBM Research Report RZ 2974, Zurich Research Laboratory, Zurich, November 1997. An extended abstract appears in K. Nyberg, ed., *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16, Springer-Verlag, 1998.
46. Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In H. Imai and Y. Zheng, eds., *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134, Springer-Verlag, 1998.
47. Yuliang Zheng and Jennifer Seberry. Immunizing public-key cryptosystems against chosen ciphertext attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, 1993.

Appendix: A Comment from EPOC Authors

As described in this manuscript and [A7], the initial version of EPOC [A5] had an error in the description; hence the current version of EPOC [A6] already includes the fix and so is proof against JQY attack.

The reason why the initial version was weak against chosen-ciphertext attack such as JQY attack is that it was not an exact implementation of [A1, A2]. In other words, the weakness of the initial version is due to the gap between the implementation [A5] and the theoretical results [A1, A2].

In [A1, A2], we have shown two different conversions from an (arbitrary) asymmetric encryption scheme, which is secure in a weaker sense, into an asymmetric encryption scheme that is secure against adaptive chosen-ciphertext attacks in the random oracle model: For message $m \in \{0,1\}^{mlen}$, picking random

string $r \in \{0,1\}^{rlen}$, the schemes obtained by the conversions are

$$\mathcal{E}_{pk}^{\mathrm{FO1}}(m;r) = \quad \mathcal{E}_{pk}^{\mathrm{asym}}((m||r); H(m,r)), \text{ and} \tag{1}$$

$$\mathcal{E}_{pk}^{\mathrm{FO2}}(m;r) = \quad \mathcal{E}_{pk}^{\mathrm{asym}}(r; H(m,r)) \quad || \quad m \oplus G(r), \tag{2}$$

respectively, where G, H denote hash functions such that $G : \{0,1\}^{rlen} \to \{0,1\}^{glen}$ and $H : \{0,1\}^{mlen} \times \{0,1\}^{rlen} \to \{0,1\}^{hlen}$. To appropriately quote from [A1, A2], the hash functions in the conversions must be carefully implemented. H in conversions, (1) and (2), should be considered as the different hash functions with the different domains. We denote by $\mathtt{MSP}$ the message space of the underlying encryption, $\mathcal{E}_{pk}^{\mathrm{asym}}$; that is, for $\mathcal{E}_{pk}^{\mathrm{asym}}(X;R)$, $X \in \mathtt{MSP}$. Following [A1, A2], it is required that $\mathtt{MSP} = `\{0,1\}^{mlen} \times \{0,1\}^{rlen}$' in EPOC-1 and $\mathtt{MSP} = `\{0,1\}^{rlen}$' [1] (The reader should not confuse $\mathtt{MSP}$ of $\mathcal{E}_{pk}^{\mathrm{asym}}$ with the *real* message space, $\{0,1\}^{mlen}$, of $\mathcal{E}_{pk}^{\mathrm{FO1}}$ and $\mathcal{E}_{pk}^{\mathrm{FO2}}$). The above requirement implies that the hash functions will halt if they take an element outside their domains (because the input is not defined!) and the decryption must abort (and output an *invalid* signal) if the hash functions invoked takes such an invalid element.

In the initial version of EPOC, H was described as a function in both conversions *carelessly* with an *inappropriate* domain such that $H : \{0,1\}^* \to \{0,1\}^{hlen}$. As mentioned later, the message space of the Okamoto-Uchiyama encryption scheme, which is used as the underlying encryption scheme in EPOC, is not equivalent to $\{0,1\}^*$: i.e., $\mathtt{MSP} \subsetneq \{0,1\}^*$. That is why the initial version was open to JQY attack — Actually, a knowledge extractor constructed by following [A1, A2] doesn't work on these wrong implementations; so the chosen-cipher security of these schemes is not guaranteed in general.

Recall the Okamoto-Uchiyama encryption scheme [A4]. For $x \in \{0,1\}^K$, picking a random string r from an appropriate domain, the encryption of x is

$$\mathcal{E}_{pk}^{\mathrm{asym}}(x;r) = \quad g^x h^r \bmod n. \tag{3}$$

Following [A1, A2], we must implement H so that $H : \{0,1\}^{mlen} \times \{0,1\}^{rlen} \to \{0,1\}^{hlen}$, where $K = mlen + rlen$ in EPOC-1 and $K = rlen$ in EPOC-2. In addition, as the Okamoto-Uchiyama scheme is an encryption scheme, we naturally get $K < |p|$, because an encryption scheme is required to satisfy the condition that, for any $x \in \mathtt{MSP}$ and $y \leftarrow E_{pk}^{\mathrm{asym}}(x)$, then $D_{sk}^{\mathrm{asym}}(y) = x$ (See [A1, A2]). If $|p| \leq K$, this condition does not hold.

As a result, an appropriate implementation wouldn't be open to any chosen-ciphertext attacks, not just JQY attack. Please refer to [A3, A6] for more details.

Finally, we would like to thank M. Joye, J.J. Quisquater, and M. Yung for giving us to place a comment in the appendix of their paper.

References

[A1] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology — CRYPTO'99*,

[1] This means that the encoding from $\mathtt{MSP}$ to $`\{0,1\}^{mlen} \times \{0,1\}^{rlen}$' is bijective in (1) and the encoding from $\mathtt{MSP}$ to $`\{0,1\}^{rlen}$' is bijective in (2).

volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.

[A2] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. *IEICE Transaction of Fundamentals of electronic Communications and Computer Science*, E83-A(1):24–32, January 2000.

[A3] E. Fujisaki and T. Okamoto. A Chosen-Cipher Secure Encryption Scheme Tightly As Secure As Factoring. *IEICE Transaction of Fundamentals of electronic Communications and Computer Science*, E84-A(1), To appear in January 2001.

[A4] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, Lecture Notes in Computer Science, pages 308–318. Springer-Verlag, 1998.

[A5] T. Okamoto, S. Uchiyama, and E. Fujisaki. EPOC: Efficient probabilistic public-key encryption. Proposal to IEEE P1363a, November 1998.

[A6] T. Okamoto, S. Uchiyama, and E. Fujisaki. EPOC: Efficient probabilistic public-key encryption. Proposal to IEEE P1363a, ver. D6, Nov. 2000, available via: `http://grouper.ieee.org/groups/1363/P1363a/draft.html`

[A7] M. Joye, J.J. Quisquater, and M. Yung. On the power of misbehaving adversaries and security analysis of EPOC. Manuscript, February 2000.

A Practical and Provably Secure Coalition-Resistant Group Signature Scheme

[Published in M. Bellare, Ed., *Advances in Cryptology – CRYPTO 2000*, vol. 1880 of *Lecture Notes in Computer Science*, pp. 255–270, Springer-Verlag, 2000.]

Giuseppe Ateniese[1], Jan Camenisch[2], Marc Joye[3], and Gene Tsudik[4]

[1] Department of Computer Science, The Johns Hopkins University
3400 North Charles Street, Baltimore, MD 21218, USA
ateniese@cs.jhu.edu
[2] IBM Research, Zurich Research Laboratory
Säumertrasse 4, CH-8803 Rüschlikon, Switzerland
jca@zurich.ibm.com
[3] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, F-13881 Gémenos, France
marc.joye@gemplus.com
[4] Department of Information and Computer Science,
University of California, Irvine, Irvine, CA 92697-3425, USA
gts@ics.uci.edu

Abstract. A group signature scheme allows a group member to sign messages anonymously on behalf of the group. However, in the case of a dispute, the identity of a signature's originator can be revealed (only) by a designated entity. The interactive counterparts of group signatures are identity escrow schemes or group identification scheme with revocable anonymity. This work introduces a new provably secure group signature and a companion identity escrow scheme that are significantly more efficient than the state of the art. In its interactive, identity escrow form, our scheme is proven secure and coalition-resistant under the strong RSA and the decisional Diffie-Hellman assumptions. The security of the non-interactive variant, i.e., the group signature scheme, relies additionally on the Fiat-Shamir heuristic (also known as the random oracle model).

Keywords. Group signature schemes, revocable anonymity, coalition-resistance, strong RSA assumption, identity escrow, provable security.

1 Introduction

Group signature schemes are a relatively recent cryptographic concept introduced by Chaum and van Heyst [CvH91] in 1991. In contrast to ordinary signatures they provide anonymity to the signer, i.e., a verifier can only tell that a member of some group signed. However, in exceptional cases such as a legal dispute, any group signature can be "opened" by a designated group manager

to reveal unambiguously the identity of the signature's originator. At the same time, no one — including the group manager — can misattribute a valid group signature.

The salient features of group signatures make them attractive for many specialized applications, such as voting and bidding. They can, for example, be used in invitations to submit tenders [CP95]. All companies submitting a tender form a group and each company signs its tender anonymously using the group signature. Once the preferred tender is selected, the winner can be traced while the other bidders remain anonymous. More generally, group signatures can be used to conceal organizational structures, e.g., when a company or a government agency issues a signed statement. Group signatures can also be integrated with an electronic cash system whereby several banks can securely distribute anonymous and untraceable e-cash. This offers concealing of the cash-issuing banks' identities [LR98].

A concept dual to group signature schemes is identity escrow [KP98]. It can be regarded as a group-member identification scheme with revocable anonymity. A group signature scheme can be turned into an identity escrow scheme by signing a random message and then proving the knowledge of a group signature on the chosen message. An identity escrow scheme can be turned into a group signature scheme using the Fiat-Shamir heuristic [FS87]. In fact, most group signature schemes are obtained in that way from 3-move honest-verifier proof of knowledge protocols.

This paper presents a new group signature / identity escrow scheme that is provably secure. In particular, the escrow identity scheme is provably coalition-resistant under the strong RSA assumption. Other security properties hold under the decisional Diffie-Hellman or the discrete logarithm assumption. Our group signature scheme is obtained from the identity escrow scheme using the Fiat-Shamir heuristic, hence it is secure in the random oracle model.

Our new (group signature) scheme improves on the state-of-the-art exemplified by the scheme of Camenisch and Michels [CM98a] which is the only known scheme whose coalition-resistance is provable under a standard cryptographic assumption. In particular, our scheme's registration protocol (`JOIN`) for new members is an order of magnitude more efficient. Moreover, our registration protocol is statistically zero-knowledge with respect to the group member's secrets. In contrast, in [CM98a] the group member is required to send the group manager the product of her secret, a prime of special form, and a random prime; such products are in principle susceptible to an attack due to Coppersmith [Cop96]. Moreover, our scheme is provably coalition-resistance against an adaptive adversary, whereas for the scheme by Camenisch and Michels [CM98a] this holds only for a static adversary.

The rest of this paper is organized as follows. The next section presents the formal model of a secure group signature scheme. Section 3 overviews cryptographic assumptions underlying the security of our scheme and introduces some basic building blocks. Subsequently, Section 4 presents the new group signature scheme. The new scheme is briefly contrasted with prior work in Section 5. The security properties are considered in Section 6. Finally, the paper concludes in Section 7.

2 The Model

Group-signature schemes are defined as follows. (For an in-depth discussion on this subject, we refer the reader to [Cam98].)

Definition 1. *A* group-signature scheme *is a digital signature scheme comprised of the following five procedures:*

`SETUP`: *On input a security parameter ℓ, this probabilistic algorithm outputs the initial group public key $\mathcal{Y}$ (including all system parameters) and the secret key $\mathcal{S}$ for the group manager.*

`JOIN`: *A protocol between the group manager and a user that results in the user becoming a new group member. The user's output is a membership certificate and a membership secret.*

`SIGN`: *A probabilistic algorithm that on input a group public key, a membership certificate, a membership secret, and a message m outputs group signature of m.*

`VERIFY`: *An algorithm for establishing the validity of an alleged group signature of a message with respect to a group public key.*

`OPEN`: *An algorithm that, given a message, a valid group signature on it, a group public key and a group manager's secret key, determines the identity of the signer.*

A secure group signature scheme must satisfy the following properties:

Correctness: Signatures produced by a group member using `SIGN` must be accepted by `VERIFY`.

Unforgeability: Only group members are able to sign messages on behalf of the group.

Anonymity: Given a valid signature of some message, identifying the actual signer is computationally hard for everyone but the group manager.

Unlinkability: Deciding whether two different valid signatures were computed by the same group member is computationally hard.

Exculpability: Neither a group member nor the group manager can sign on behalf of other group members.[1] A closely related property is that of *nonframing* [CP95]; it captures the notion of a group member not being made responsible for a signature she did not produce.

Traceability: The group manager is always able to open a valid signature and identify the actual signer.

Note that the last property is also violated if a subset of group members, pooling together their secrets, can generate a valid group signature that cannot be opened by the group manager. Because this was ignored in many papers we state it explicitly as an additional property.

[1] Note that the above does not preclude the group manager from creating fraudulent signers (i.e., nonexistent group members) and then producing group signatures.

Coalition-resistance: A colluding subset of group members (even if comprised of the entire group) cannot generate a valid signature that the group manager cannot link to one of the colluding group members.

We observe that many group signature schemes (e.g., [CM98a, CM99b, CS97]) can be viewed as making use of two different ordinary signature schemes: one to generate membership certificates as part of `JOIN` and another to actually generate group signatures as part of `SIGN` (cf. [CM99b]). Consequently, the properties of any secure group signature scheme must include the Unforgeability property (as defined in [GMR88]) for each of the two ordinary signature schemes. It is easy to see that each of: Traceability and Exculpability map into the Unforgeability property for the two respective signature schemes. Furthermore, together they ensure that a group signature scheme is unforgeable, i.e., that only group members are able to sign messages on behalf of the group.

The model of identity escrow schemes [KP98] is basically the same as the one for group signature schemes; the only difference being that the `SIGN` algorithm is replaced by an interactive protocol between a group member and a verifier.

3 Preliminaries

This section reviews some cryptographic assumptions and introduces the building blocks necessary in the subsequent design of our group signature scheme. (It can be skipped with no significant loss of continuity.)

3.1 Number-theoretic assumptions

The *Strong-RSA Assumption* (SRSA) was independently introduced by Barić and Pfitzmann [BF97] and by Fujisaki and Okamoto [FO97]. It strengthens the widely accepted RSA Assumption that finding e^{th}-roots modulo n — where e is the public, and thus fixed, exponent — is hard to the assumption that finding an e^{th}-root modulo n for any $e > 1$ is hard. We give hereafter a more formal definition.

Definition 2 (Strong-RSA Problem). *Let $n = pq$ be an RSA-like modulus and let G be a cyclic subgroup of $\mathbb{Z}_n^*$ of order $\#G$, $\lceil \log_2(\#G) \rceil = \ell_G$. Given n and $z \in G$, the* Strong-RSA Problem *consists in finding $u \in G$ and $e \in \mathbb{Z}_{>1}$ satisfying $z \equiv u^e \pmod{n}$.*

Assumption 1 (Strong-RSA Assumption). *There exists a probabilistic polynomial-time algorithm $\mathcal{K}$ which on input a security parameter ℓ_G outputs a pair (n, z) such that, for all probabilistic polynomial-time algorithms $\mathcal{P}$, the probability that $\mathcal{P}$ can solve the Strong-RSA Problem is negligible.*

The *Diffie-Hellman Assumption* [DH76] appears in two "flavors": (i) the Computational Diffie-Hellman Assumption (CDH) and (ii) the Decisional Diffie-Hellman Assumption (DDH). For a thorough discussion on the subject we refer the reader to [Bon98].

Definition 3 (Decisional Diffie-Hellman Problem). *Let $G = \langle g \rangle$ be a cyclic group generated by g of order $u = \#G$ with $\lceil \log_2(u) \rceil = \ell_G$. Given g, g^x, g^y, and $g^z \in G$, the* Decisional Diffie-Hellman Problem *consists in deciding whether the elements g^{xy} and g^z are equal.*

This problem gives rise to the *Decisional Diffie-Hellman Assumption*, which was first explicitly mentioned in [Bra93] by Brands although it was already implicitly assumed in earlier cryptographic schemes.

Assumption 2 (Decisional Diffie-Hellman Assumption). *There is no probabilistic polynomial-time algorithm that distinguishes with non-negligible probability between the distributions D and R, where $D = (g, g^x, g^y, g^z)$ with $x, y, z \in_R \mathbb{Z}_u$ and $R = (g, g^x, g^y, g^{xy})$ with $x, y \in_R \mathbb{Z}_u$.*

The Decisional Diffie-Hellman Problem is easier than the *(Computational) Diffie-Hellman Problem* which involves finding g^{uv} from g^u and g^v; the Decisional Diffie-Hellman Assumption is, thus, a *stronger* assumption. Both are stronger assumptions than the assumption that computing discrete logarithms is hard.

If n is a *safe* RSA modulus (i.e., $n = pq$ with $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are all prime), it is a good habit to restrict operation to the subgroup of quadratic residues modulo n, i.e., the cyclic subgroup $\mathrm{QR}(n)$ generated by an element of order $p'q'$. This is because the order $p'q'$ of $\mathrm{QR}(n)$ has no small factors.

The next corollary shows that it is easy to find a generator g of $\mathrm{QR}(n)$: it suffices to choose an element $a \in \mathbb{Z}_n^*$ satisfying $\gcd(a \pm 1, n) = 1$ and then to take $g = a^2 \bmod n$. We then have $\mathrm{QR}(n) = \langle g \rangle$. (By convention, $\gcd(0, n) := n$.)

Proposition 1. *Let $n = pq$, where $p \neq q$, $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are all prime. The order of the elements in $\mathbb{Z}_n^*$ are one of the set $\{1, 2, p', q', 2p', 2q', p'q', 2p'q'\}$. Moreover, if the order of $a \in \mathbb{Z}_n^*$ is equal to $p'q'$ or $2p'q'$ $\iff$ $\gcd(a \pm 1, n) = 1$.* □

Corollary 1. *Let n be as in Proposition 1. Then, for any $a \in \mathbb{Z}_n^*$ s.t. $\gcd(a \pm 1, n) = 1$, $\langle a^2 \rangle \subset \mathbb{Z}_n^*$ is a cyclic subgroup of order $p'q'$.* □

Remark 1. Notice that $4\ (= 2^2)$ always generates $\mathrm{QR}(n)$ whatever the value of a safe RSA modulus n. Notice also that the Jacobi symbol $(g|n) = +1$ does *not* necessarily imply that g is a quadratic residue modulo n but merely that $(g|p) = (g|q) = \pm 1$, where $(g|p)$ (resp. $(g|q)$) denotes the Legendre symbol[2] of g modulo p (resp. q). For example, $(2|55) = (2|5)\,(2|11) = (-1)\,(-1) = +1$; however, there is no integer x such that $x^2 \equiv 2 \pmod{55}$.

Deciding whether some y is in $\mathrm{QR}(n)$ is generally believed infeasible if the factorization of n is unknown.

[2] By definition, the Legendre symbol $(g|p) = +1$ if g is a quadratic residue modulo p, and -1 otherwise.

3.2 Signatures of knowledge

So-called zero-knowledge proofs of knowledge allow a prover to demonstrate the knowledge of a secret w.r.t. some public information such that no other information is revealed in the process. The protocols we use in the following are all 3-move protocols and can be proven zero-knowledge in an honest-verifier model. Such protocols can be performed non-interactively with the help of an ideal hash function $\mathcal{H}$ (*à la* Fiat-Shamir [FS87]). Following [CS97], we refer to the resulting constructs as *signatures of knowledge*. One example is the Schnorr signature scheme [Sch91] where a signature can be viewed as a proof of knowledge of the discrete logarithm of the signer's public key made non-interactive.

In the following, we consider three building blocks: signature of knowledge of (i) a discrete logarithm; (ii) equality of two discrete logarithms; and (iii) a discrete logarithm lying in a given interval. All of these are constructed over a cyclic group $G = \langle g \rangle$ the order of which $\#G$ is unknown; however its bit-length ℓ_G (i.e., the integer ℓ_G s.t. $2^{\ell_G - 1} \leq \#G < 2^{\ell_G}$) is publicly known. Fujisaki and Okamota [FO97] show that, under the SRSA, the standard proofs of knowledge protocols that work for a group of known order are also proofs of knowledge in this setting. We define the *discrete logarithm* of $y \in G$ w.r.t. base g as any integer $x \in \mathbb{Z}$ such that $y = g^x$ in G. We denote $x = \log_g y$. We assume a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ which maps a binary string of arbitrary length to a k-bit hash value. We also assume a security parameter $\epsilon > 1$.

Showing the knowledge of the discrete logarithm of $y = g^x$ can be done easily in this setting as stated by the following definition (cf. [Sch91]).

Definition 4. *Let $y, g \in G$. A pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\epsilon(\ell_G + k)+1}$ verifying $c = \mathcal{H}(y \| g \| g^s\, y^c \| m)$ is a signature of knowledge of the discrete logarithm of $y = g^x$ w.r.t. base g, on a message $m \in \{0,1\}^*$.*

The party in possession of the secret $x = \log_g y$ is able to compute the signature by choosing a random $t \in \pm\{0,1\}^{\epsilon(\ell_G + k)}$ and then computing c and s as:

$$c = \mathcal{H}(y \| g \| g^t \| m) \quad \text{and} \quad s = t - cx \quad (\text{in } \mathbb{Z}) \ .$$

A slight modification of the previous definition enables to show the knowledge and equality of two discrete logarithms of, say y_1 and y_2, with bases g and h, i.e., knowledge of an integer x satisfying $y_1 = g^x$ and $y_2 = h^x$.

Definition 5. *Let $y_1, y_2, g, h \in G$. A pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\epsilon(\ell_G + k)+1}$ verifying $c = \mathcal{H}(y_1 \| y_2 \| g \| h \| g^s\, {y_1}^c \| h^s\, {y_2}^c \| m)$ is a signature of knowledge of the discrete logarithm of both $y_1 = g^x$ w.r.t. base g and $y_2 = h^x$ w.r.t. base h, on a message $m \in \{0,1\}^*$.*

The party in possession of the secret x is able to compute the signature, provided that $x = \log_g y_1 = \log_h y_2$, by choosing a random $t \in \pm\{0,1\}^{\epsilon(\ell_G + k)}$ and then computing c and s as:

$$c = \mathcal{H}(y_1 \| y_2 \| g \| h \| g^t \| h^t \| m) \quad \text{and} \quad s = t - cx \quad (\text{in } \mathbb{Z}) \ .$$

In Definition 4, a party shows the knowledge of the discrete logarithm of y w.r.t. base g. The order of g being unknown, this means that this party knows an integer x satisfying $y = g^x$. This latter condition may be completed in the sense that the party knows a discrete logarithm x lying in a given interval. It is a slight modification of a protocol appearing in [FO98].

Definition 6. *Let $y, g \in G$. A pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\epsilon(\ell+k)+1}$ verifying $c = \mathcal{H}(y \,\|g\, \|g^{s-cX}\, y^c \,\|m)$ is a signature of knowledge of the discrete logarithm $\log_g y$ that lies in $]X - 2^{\epsilon(\ell+k)}, X + 2^{\epsilon(\ell+k)}[$, on a message $m \in \{0,1\}^*$.*

From the knowledge of $x = \log_g y \in]X - 2^\ell, X + 2^\ell[$, this signature is obtained by choosing a random $t \in \pm\{0,1\}^{\epsilon(\ell+k)}$ and computing c and s as:

$$c = \mathcal{H}(y\|g\|g^t\|m),\ s = t - c(x - X) \quad (\text{in } \mathbb{Z}) \ .$$

Remark 2. Note that, although the party knows a secret x in $]X-2^\ell, X+2^\ell[$, the signature only guarantees that x lies in the extended interval $]X - 2^{\epsilon(\ell+k)}, X + 2^{\epsilon(\ell+k)}[$.

The security of all these building blocks has been proven in the random oracle model [BR93] under the strong RSA assumption in [CM98b, FO97, FO98]. That is, if $\epsilon > 1$, then the corresponding interactive protocols are statistical (honest-verifier) zero-knowledge proofs of knowledge.

4 The New Group Signature and Identity Escrow Schemes

This section describes our new group signature scheme and tells how an identity escrow scheme can be derived.

As mentioned in Section 2, many recent group signature schemes involve applying two types of non-group signature schemes: one for issuing certificates and one for actual group-signatures, respectively. The security of the former, in particular, is of immediate relevance because it assures, among other things, the coalition-resistance property of a group signature scheme. The reasoning for this assertion is fairly intuitive:

> Each group member obtains a unique certificate from the group manager as part of `JOIN` where each certificate is actually a signature over a secret random message chosen by each member. As a coalition, all group members can be collectively thought of as a single adversary mounting an adaptive chosen message attack consisting of polynomially many instances of `JOIN`.

The main challenge in designing a practical group signature scheme is in finding a signature scheme for the certification of membership that allows the second signature scheme (which is used to produce actual group signatures) to remain efficient. Typically, the second scheme is derived (using the Fiat-Shamir heuristic) from a proof of knowledge of a membership certificate. Hence, the

certification signature scheme must be such that the latter proof can be realized efficiently.

Recall that proving knowledge of a hash function pre-image is, in general, not possible in an efficient manner. Therefore, a candidate signature scheme must replace the hash function with another suitable function. However, because `JOIN` is an interactive protocol between the new member and the group manager, the latter can limit and influence what he signs (e.g., assure that it signs a *random* message).

4.1 The group signature scheme

Let $\epsilon > 1$, k, and ℓ_p be security parameters and let λ_1, λ_2, γ_1, and γ_2 denote lengths satisfying $\lambda_1 > \epsilon(\lambda_2+k)+2$, $\lambda_2 > 4\ell_p$, $\gamma_1 > \epsilon(\gamma_2+k)+2$, and $\gamma_2 > \lambda_1+2$. Define the integral ranges $\Lambda =]2^{\lambda_1} - 2^{\lambda_2}, 2^{\lambda_1} + 2^{\lambda_2}[$ and $\Gamma =]2^{\gamma_1} - 2^{\gamma_2}, 2^{\gamma_1} + 2^{\gamma_2}[$. Finally, let $\mathcal{H}$ be a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$. (The parameter ϵ controls the tightness of the statistical zero-knowledgeness and the parameter ℓ_p sets the size of the modulus to use.)

The initial phase involves the group manager (GM) setting the group public and his secret keys: $\mathcal{Y}$ and $\mathcal{S}$.

`SETUP:`

1. Select random secret ℓ_p-bit primes p', q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are prime. Set the modulus $n = pq$.
2. Choose random elements $a, a_0, g, h \in_R \mathrm{QR}(n)$ (of order $p'q'$).
3. Choose a random secret element $x \in_R \mathbb{Z}^*_{p'q'}$ and set $y = g^x \bmod n$.
4. The group public key is: $\mathcal{Y} = (n, a, a_0, y, g, h)$.
5. The corresponding secret key (known only to GM) is: $\mathcal{S} = (p', q', x)$.

Remark 3. The group public key $\mathcal{Y}$ is made available via the usual means (i.e., embedded in some form of a public key certificate signed by a trusted authority). We note that, in practice, components of $\mathcal{Y}$ must be verifiable to prevent framing attacks. In particular, Proposition 1 provides an efficient way to test whether an element has order at least $p'q'$. Then it is sufficient to square this element to make sure it is in $\mathrm{QR}(n)$, with order $p'q'$. GM also needs to provide a proof that n is the product of two safe primes ([CM99a] shows how this can be done).

Suppose now that a new user wants to join the group. We assume that communication between the user and the group manager is secure, i.e., private and authentic. The selection of per-user parameters is done as follows:

`JOIN:`

1. User P_i generates a secret exponent $\tilde{x}_i \in_R]0, n^2[$, a random integer $\tilde{r} \in_R]0, 2^{2\ell_p}[$ and sends $C_1 = g^{\tilde{x}_i} h^{\tilde{r}} \bmod n$ to GM and proves him knowledge of the representation of C_1 w.r.t. bases g and h.
2. GM checks that $C_1 \in \mathrm{QR}(n)$. If this is the case, GM selects α_i and $\beta_i \in_R]0, 2^{\lambda_2}[$ at random and sends (α_i, β_i) to P_i.

3. User P_i computes $x_i = 2^{\lambda_1} + (\alpha_i \tilde{x}_i + \beta_i \bmod 2^{\lambda_2})$ and sends GM the value $C_2 = a^{x_i} \bmod n$. The user also proves to GM:
   (a) that the discrete log of C_2 w.r.t. base a lies in Λ, and
   (b) knowledge of integers u, v, and w such that
      i. u lies in $]-2^{\lambda_2}, 2^{\lambda_2}[$,
      ii. u equals the discrete log of $C_2/a^{2^{\lambda_1}}$ w.r.t. base a, and
      iii. $C_1^{\alpha_i} g^{\beta_i}$ equals $g^u (g^{2^{\lambda_2}})^v h^w$ (see Definition 6).

   (The statements (i–iii) prove that the user's membership secret $x_i = \log_a C_2$ is correctly computed from C_1, α_i, and β_i.)
4. GM checks that $C_2 \in \mathrm{QR}(n)$. If this is the case and all the above proofs were correct, GM selects a random prime $e_i \in_R \Gamma$ and computes $A_i := (C_2\, a_0)^{1/e_i} \bmod n$. Finally, GM sends P_i the new membership certificate $[A_i, e_i]$. (Note that $A_i = (a^{x_i}\, a_0)^{1/e_i} \bmod n$.)
5. User P_i verifies that $a^{x_i} a_0 \equiv {A_i}^{e_i} \pmod n$.

Remark 4. As part of `JOIN`, GM creates a new entry in the membership table and stores $\{[A_i, e_i]$, `JOIN` transcript$\}$ in the new entry. (`JOIN` transcript is formed by the messages received from and sent to the user in the steps above. It is assumed to be signed by the user with some form of a long-term credential.)

Armed with a membership certificate $[A_i, e_i]$, a group member can generate anonymous and unlinkable group signatures on a generic message $m \in \{0,1\}^*$:

`SIGN:`

1. Generate a random value $w \in_R \{0,1\}^{2\ell_p}$ and compute:

$$T_1 = A_i\, y^w \bmod n, \quad T_2 = g^w \bmod n, \quad T_3 = g^{e_i}\, h^w \bmod n \ .$$

2. Randomly choose $r_1 \in_R \pm\{0,1\}^{\epsilon(\gamma_2+k)}$, $r_2 \in_R \pm\{0,1\}^{\epsilon(\lambda_2+k)}$, $r_3 \in_R \pm\{0,1\}^{\epsilon(\gamma_1+2\ell_p+k+1)}$, and $r_4 \in_R \pm\{0,1\}^{\epsilon(2\ell_p+k)}$ and compute:
   (a) $d_1 = {T_1}^{r_1}/(a^{r_2}\, y^{r_3}) \bmod n$, $d_2 = {T_2}^{r_1}/g^{r_3} \bmod n$, $d_3 = g^{r_4} \bmod n$, and $d_4 = g^{r_1}\, h^{r_4} \bmod n$;
   (b) $c = \mathcal{H}(g \| h \| y \| a_0 \| a \| T_1 \| T_2 \| T_3 \| d_1 \| d_2 \| d_3 \| d_4 \| m)$;
   (c) $s_1 = r_1 - c(e_i - 2^{\gamma_1})$, $s_2 = r_2 - c(x_i - 2^{\lambda_1})$, $s_3 = r_3 - c\, e_i\, w$, and $s_4 = r_4 - c\, w$ (all in $\mathbb{Z}$).
3. Output $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$.

A group signature can be regarded as a signature of knowledge of (1) a value $x_i \in \Lambda$ such that $a^{x_i} a_0$ is the value that is ElGamal-encrypted in (T_1, T_2) under y and of (2) an e_i-th root of that encrypted value, where e_i is the first part of the representation of T_3 w.r.t. g and h and that e_i lies in Γ.

A verifier can check the validity of a signature $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$ of the message m as follows.

`VERIFY:`

1. Compute:

$$c' = \mathcal{H}\big(g\|h\|y\|a_0\|a\|T_1\|T_2\|T_3\|{a_0}^c\,{T_1}^{s_1-c2^{\gamma_1}}/(a^{s_2-c2^{\lambda_1}}\,y^{s_3}) \bmod n\,\|\; {T_2}^{s_1-c2^{\gamma_1}}/g^{s_3} \bmod n\,\|{T_2}^c\,g^{s_4} \bmod n\,\|{T_3}^c\,g^{s_1-c2^{\gamma_1}}\,h^{s_4} \bmod n\,\|\,m\big)\ .$$

2. Accept the signature if and only if $c = c'$, and $s_1 \in \pm\{0,1\}^{\epsilon(\gamma_2+k)+1}$, $s_2 \in \pm\{0,1\}^{\epsilon(\lambda_2+k)+1}$, $s_3 \in \pm\{0,1\}^{\epsilon(\lambda_1+2\ell_p+k+1)+1}$, $s_4 \in \pm\{0,1\}^{\epsilon(2\ell_p+k)+1}$.

In the event that the actual signer must be subsequently identified (e.g., in case of a dispute) GM executes the following procedure:

`OPEN:`

1. Check the signature's validity via the `VERIFY` procedure.
2. Recover A_i (and thus the identity of P_i) as $A_i = T_1/{T_2}^x \bmod n$.
3. Prove that $\log_g y = \log_{T_2}(T_1/A_i \bmod n)$ (see Definition 5).

4.2 Deriving an identity escrow scheme

Only minor changes are necessary to construct an identity escrow scheme out of the proposed group signature scheme. Specifically, the `SIGN` and `VERIFY` procedures must be replaced by an interactive protocol between a group member (prover) and a verifier. This protocol can be derived from `SIGN` by replacing the call to the hash function $\mathcal{H}$ by a call to the verifier. That is, the prover sends to the verifier all inputs to the hash function $\mathcal{H}$ and gets back a value $c \in \{0,1\}^{\ell_c}$ randomly chosen by the verifier, with $\ell_c = O(\log \ell_p)$. Then, the prover computes the s_i's and sends these back to the verifier. The verification equation that the verifier uses to check can be derived from the argument to $\mathcal{H}$ in `VERIFY`. Depending on the choice of the security parameters, the resulting protocol must be repeated sufficiently many times to obtain a small enough probability of error.

5 Related Work

Previously proposed group signature schemes can be divided into two classes: (I) schemes where the sizes of the group public key and/or of group signatures (linearly) depend on the number of group members and (II) schemes where the sizes of the group public key and of group signatures are constant. Most of the early schemes belong to the first class. Although many of those have been proven secure with respect to some standard cryptographic assumption (such as the hardness of computing discrete logarithms) they are inefficient for large groups.

Numerous schemes of Class II have been proposed, however, most are either insecure (or of dubious security) or are grossly inefficient. The only notable and efficient group signature scheme is due to Camenisch and Michels [CM98b].

Our scheme differs from the Camenisch/Michels scheme mainly in the membership certificate format. As a consequence, our `JOIN` protocol has two important advantages:

(1) Our `JOIN` protocol is an order of magnitude more efficient since all proofs that the new group member must provide are efficient proofs of knowledge of discrete logarithms. This is in contrast to the Camenisch/Michels scheme where the group member must prove that some number is the product of two primes. The latter can be realized only with binary challenges.
(2) Our `JOIN` protocol is more secure for the group members, i.e., it is statistical zero-knowledge with respect to the group member's membership secret. The `JOIN` protocol in the Camenisch-Michels scheme is not; in fact, it requires the group member to expose the product of her secret, a prime of special form, and a random prime; such products are in principle susceptible to an attack due to Coppersmith [Cop96]. (Although, the parameters of their scheme can be set such that this attack becomes infeasible.)

Furthermore, the proposed scheme is provably coalition-resistant against an *adaptive* adversary. This offers an extra advantage:

(3) Camenisch and Michels prove their scheme coalition-resistant against a static adversary who is given all certificates as input, whereas our scheme can handle a much more powerful and realistic adversary that is allowed to *adaptively* run the `JOIN` protocol.

6 Security of the Proposed Schemes

In this section we assess the security of the new group signature scheme and the companion escrow identity scheme. We first need to prove that the following theorems hold.

Theorem 1 (Coalition-resistance). *Under the strong RSA assumption, a group certificate* $[A_i = (a^{x_i}\, a_0)^{1/e_i} \bmod n, e_i]$ *with* $x_i \in \Lambda$ *and* $e_i \in \Gamma$ *can be generated only by the group manager provided that the number* K *of certificates the group manager issues is polynomially bounded.*

Proof. Let $\mathcal{M}$ be an attacker that is allowed to adaptively run the `JOIN` and thereby obtain group certificates $[A_j = (a^{x_j}\, a_0)^{1/e_j} \bmod n, e_j]$, $j = 1, \ldots, K$. Our task is now to show that if $\mathcal{M}$ outputs a tuple $\big(\hat{x}; [\hat{A}, \hat{e}]\big)$, with $\hat{x} \in \Lambda$, $\hat{e} \in \Gamma$, $\hat{A} = (a^{\hat{x}}\, a_0)^{1/\hat{e}} \bmod n$, and $(\hat{x}, \hat{e}) \neq (x_j, e_j)$ for all $1 \leq j \leq K$ with non-negligible probability, then the strong RSA assumption does not hold.

Given a pair (n, z), we repeatedly play a random one of the following two games with $\mathcal{M}$ and hope to calculate a pair $(u, e) \in \mathbb{Z}_n^* \times \mathbb{Z}_{>1}$ satisfying $u^e \equiv z \pmod{n}$ from $\mathcal{M}$'s answers. The first game goes as follows:

1. Select $x_1, \ldots, x_K \in \Lambda$ and $e_1, \ldots, e_K \in \Gamma$.
2. Set $a = z^{\prod_{1 \leq l \leq K} e_l} \bmod n$.

3. Choose $r \in_R \Lambda$ and set $a_0 = a^r \bmod n$.
4. For all $1 \le i \le K$, compute $A_i = z^{(x_i+r)\prod_{1\le l\le K; l\neq i} e_l} \bmod n$.
5. Select $g, h \in_R \mathrm{QR}(n)$, $x \in \{1, \ldots, n^2\}$, and set $y = g^x \bmod n$.
6. Run the `JOIN` protocol K times with $\mathcal{M}$ on input (n, a, a_0, y, g, h). Assume we are in protocol run i. Receive the commitment C_1 from $\mathcal{M}$. Use the proof of knowledge of a representation of C_1 with respect to g and h to extract $\tilde{x}_i$ and $\tilde{r}_i$ such that $C_1 = g^{\tilde{x}_i} h^{\tilde{r}_i}$ (this involves rewinding of $\mathcal{M}$). Choose α_i and $\beta_i \in]0, 2^{\lambda_2}[$ such that the prepared $x_i = 2^{\lambda_1} + (\alpha_i \tilde{x}_i + \beta_i \bmod 2^{\lambda_2})$ and send α_i and β_i to $\mathcal{M}$. Run the rest of the protocol as specified until Step 4. Then send $\mathcal{M}$ the membership certificate $[A_i, e_i]$.
   After these K registration protocols are done, $\mathcal{M}$ outputs $\big(\hat{x}; [\hat{A}, \hat{e}]\big)$ with $\hat{x} \in \Lambda$, $\hat{e} \in \Gamma$, and $\hat{A} = (a^{\hat{x}} a_0)^{1/\hat{e}} \bmod n$.
7. If $\gcd(\hat{e}, e_j) \neq 1$ for all $1 \le j \le K$ then output $\perp$ and quit. Otherwise, let $\tilde{e} := (\hat{x}+r) \prod_{1\le l\le K} e_l$. (Note that $\hat{A}^{\hat{e}} \equiv z^{\tilde{e}} \pmod n$.) Because $\gcd(\hat{e}, e_j) = 1$ for all $1 \le j \le K$, we have $\gcd(\hat{e}, \tilde{e}) = \gcd(\hat{e}, (\hat{x}+r))$. Hence, by the extended Euclidean algorithm, there exist $\alpha, \beta \in \mathbb{Z}$ s.t. $\alpha\,\hat{e} + \beta\,\tilde{e} = \gcd(\hat{e}, (\hat{x}+r))$. Therefore, letting $u := z^{\alpha}\, \hat{A}^{\beta} \bmod n$ and $e := \hat{e}/\gcd(\hat{e}, (\hat{x}+r)) > 1$ because $\hat{e} > (\hat{x}+r)$, we have $u^e \equiv z \pmod n$. Output (u, e).

The previous game is only successful if $\mathcal{M}$ returns a new certificate $[A(\hat{x}), \hat{e}]$, with $\gcd(\hat{e}, e_j) = 1$ for all $1 \le j \le K$. We now present a game that solves the strong RSA problem in the other case when $\gcd(\hat{e}, e_j) \neq 1$ for some $1 \le j \le K$. (Note that $\gcd(\hat{e}, e_j) \neq 1$ means $\gcd(\hat{e}, e_j) = e_j$ because e_j is prime.)

1. Select $x_1, \ldots, x_K \in \Lambda$ and $e_1, \ldots, e_K \in \Gamma$.
2. Choose $j \in_R \{1, \ldots, K\}$ and set $a = z^{\prod_{1\le l\le K; l\neq j} e_l} \bmod n$.
3. Choose $r \in_R \Lambda$ and set $A_j = a^r \bmod n$ and $a_0 = A_j^{e_j}/a^{x_j} \bmod n$.
4. For all $1 \le i \le K$ $i \neq j$, compute $A_i = z^{(x_i + e_j r - x_j)\prod_{1\le l\le K; l\neq i,j} e_l} \bmod n$.
5. Select $g, h \in_R \mathrm{QR}(n)$, $x \in \{1, \ldots, n^2\}$, and set $y = g^x \bmod n$.
6. Run the `JOIN` protocol K times with $\mathcal{M}$ on input (n, a, a_0, y, g, h). Assume we are in protocol run i. Receive the commitment C_1 from $\mathcal{M}$. Use the proof of knowledge of a representation of C_1 with respect to g and h to extract $\tilde{x}_i$ and $\tilde{r}_i$ such that $C_1 = g^{\tilde{x}_i} h^{\tilde{r}_i} \bmod n$ (this involves rewinding of $\mathcal{M}$). Choose α_i and $\beta_i \in]0, 2^{\lambda_2}[$ such that the prepared $x_i = 2^{\lambda_1} + (\alpha_i \tilde{x}_i + \beta_i \bmod 2^{\lambda_2})$ and send α_i and β_i to $\mathcal{M}$. Run the rest of the protocol as specified until Step 4. Then send $\mathcal{M}$ the membership certificate $[A_i, e_i]$.
   After these K registration protocols are done, $\mathcal{M}$ outputs $\big(\hat{x}; [\hat{A}, \hat{e}]\big)$ with $\hat{x} \in \Lambda$, $\hat{e} \in \Gamma$, and $\hat{A} = (a^{\hat{x}} a_0)^{1/\hat{e}} \bmod n$.
7. If $\gcd(\hat{e}, e_j) \neq e_j$ output $\perp$ and quit. Otherwise, we have $\hat{e} = t\, e_j$ for some t and can define $Z := \hat{A}^t/A_j \bmod n$ if $\hat{x} \ge x_j$ and $Z := A_j/\hat{A}^t \bmod n$ otherwise. Hence, $Z \equiv (a^{|\hat{x}-x_j|})^{1/e_j} \equiv (z^{|\tilde{e}|})^{1/e_j} \pmod n$ with $\tilde{e} := (\hat{x} - x_j) \prod_{1\le l\le K; l\neq j} e_l$. Because $\gcd(e_j, \prod_{\substack{1\le l\le K \\ l\neq j}} e_l) = 1$, it follows that $\gcd(e_j, |\tilde{e}|) = \gcd(e_j, |\hat{x} - x_j|)$. Hence, there exist $\alpha, \beta \in \mathbb{Z}$ s.t. $\alpha\, e_j + \beta\, |\tilde{e}| = \gcd(e_j, |\hat{x} - x_j|)$. So, letting $u := z^{\alpha}\, Z^{\beta} \bmod n$ and $e := e_j/\gcd(e_j, |\hat{x} - x_j|) > 1$ because $e_j > |\hat{x} - x_j|$, we have $u^e \equiv z \pmod n$. Output (u, e).

Consequently, by playing randomly one of the Games 1 or 2 until the result is not $\perp$, an attacker getting access to machine $\mathcal{M}$ can solve the strong RSA problem in expected running-time polynomial in K. Because the latter is assumed to be infeasible, it follows that no one but the group manager can generate group certificates. □

Theorem 2. *Under the strong RSA assumption, the interactive protocol underlying the group signature scheme (i.e., the identification protocol of the identity escrow scheme) is a statistical zero-knowledge (honest-verifier) proof of knowledge of a membership certificate and a corresponding membership secret key.*

Proof. The proof that the interactive protocol is statistical zero-knowledge is quite standard. We restrict our attention the proof of knowledge part.

We have to show that the knowledge extractor is able to recover the group certificate once it has found two accepting tuples. Let $(T_1, T_2, T_3, d_1, d_2, d_3, d_4, c, s_1, s_2, s_3, s_4)$ and $(T_1, T_2, T_3, d_1, d_2, d_3, d_4, \tilde{c}, \tilde{s}_1, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4)$ be two accepting tuples. Since $d_3 \equiv g^{s_4} {T_2}^c \equiv g^{\tilde{s}_4} {T_2}^{\tilde{c}} \pmod{n}$, it follows that $g^{s_4-\tilde{s}_4} \equiv {T_2}^{\tilde{c}-c} \pmod{n}$. Letting $\delta_4 = \gcd(s_4 - \tilde{s}_4, \tilde{c} - c)$, by the extended Euclidean algorithm, there exist $\alpha_4, \beta_4 \in \mathbb{Z}$ s.t. $\alpha_4 (s_4 - \tilde{s}_4) + \beta_4 (\tilde{c} - c) = \delta_4$. Hence,

$$g \equiv g^{(\alpha_4 (s_4-\tilde{s}_4)+\beta_4 (\tilde{c}-c))/\delta_4} \equiv ({T_2}^{\alpha_4} g^{\beta_4})^{\frac{\tilde{c}-c}{\delta_4}} \pmod{n} \ .$$

Note that we cannot have $\tilde{c} - c < \delta_4$ because otherwise ${T_2}^{\alpha_4} g^{\beta_4}$ is a $(\frac{\tilde{c}-c}{\delta_4})^{\text{th}}$ root of g, which contradicts the strong RSA assumption. Thus, we have $\tilde{c} - c = \delta_4 = \gcd(s_4 - \tilde{s}_4, \tilde{c} - c)$; or equivalently, there exists $\tau_4 \in \mathbb{Z}$ s.t. $s_4 - \tilde{s}_4 = \tau_4(\tilde{c} - c)$. So, because $s_4 + c\,w = \tilde{s}_4 + \tilde{c}\,w$, we have $\tau_4 = w$ and thus obtain

$$A_i = \frac{T_1}{y^{\tau_4}} \bmod n \ .$$

Moreover, because $d_4 \equiv g^{s_1} h^{s_4} (T_3\, g^{-2^{\gamma_1}})^c \equiv g^{\tilde{s}_1} h^{\tilde{s}_4} (T_3\, g^{-2^{\gamma_1}})^{\tilde{c}} \pmod{n}$, we have $g^{s_1-\tilde{s}_1} \equiv (T_3\, g^{-2^{\gamma_1}})^{\tilde{c}-c} h^{\tilde{s}_4-s_4} \equiv (T_3\, g^{-2^{\gamma_1}} h^{-\tau_4})^{\tilde{c}-c} \pmod{n}$. Let $\delta_1 = \gcd(s_1 - \tilde{s}_1, \tilde{c} - c)$. By the extended Euclidean algorithm, there exist $\alpha_1, \beta_1 \in \mathbb{Z}$ s.t. $\alpha_1 (s_1 - \tilde{s}_1) + \beta_1 (\tilde{c} - c) = \delta_1$. Therefore, $g \equiv g^{(\alpha_1 (s_1-\tilde{s}_1)+\beta_1 (\tilde{c}-c))/\delta_1} \equiv [(T_3\, g^{-2^{\gamma_1}} h^{-\tau_4})^{\alpha_1} g^{\beta_1}]^{\frac{\tilde{c}-c}{\delta_1}} \pmod{n}$. This, in turn, implies by the strong RSA assumption that $\tilde{c} - c = \delta_1 = \gcd(s_1 - \tilde{s}_1, \tilde{c} - c)$; or equivalently that there exists $\tau_1 \in \mathbb{Z}$ s.t. $s_1 - \tilde{s}_1 = \tau_1(\tilde{c} - c)$. Consequently, because $s_1 + c(e_i - 2^{\gamma_1}) = \tilde{s} + \tilde{c}(e_i - 2^{\gamma_1})$, we find

$$e_i = 2^{\gamma_1} + \tau_1 \ .$$

Likewise, from $d_2 \equiv {T_2}^{s_1} g^{-s_3} ({T_2}^{-2^{\gamma_1}})^c \equiv {T_2}^{\tilde{s}_1} g^{-\tilde{s}_3} ({T_2}^{-2^{\gamma_1}})^{\tilde{c}} \pmod{n}$, it follows that $g^{s_3-\tilde{s}_3} \equiv ({T_2}^{\tau_1+2^{\gamma_1}})^{\tilde{c}-c} \pmod{n}$. Therefore, by the extended Euclidean algorithm, we can conclude that there exists $\tau_3 \in \mathbb{Z}$ s.t. $s_3 - \tilde{s}_3 = \tau_3(\tilde{c} - c)$. Finally, from $d_1 \equiv {T_1}^{s_1} a^{-s_2} y^{-s_3} ({T_1}^{-2^{\gamma_1}} a^{2^{\lambda_1}} a_0)^c \equiv {T_1}^{\tilde{s}_1} a^{-\tilde{s}_2} y^{-\tilde{s}_3} ({T_1}^{-2^{\gamma_1}} a^{2^{\lambda_1}} a_0)^{\tilde{c}} \pmod{n}$, we obtain $a^{\tilde{s}_2-s_2} \equiv ({T_1}^{\tau_1+2^{\gamma_1}} y^{-\tau_3} a^{-2^{\lambda_1}} {a_0}^{-1})^{\tilde{c}-c} \pmod{n}$ and similarly conclude that there exists $\tau_2 \in \mathbb{Z}$ s.t. $s_2 - \tilde{s}_2 = \tau_2(\tilde{c} - c)$. Because $s_2 + c(x_i - 2^{\lambda_1}) = \tilde{s}_2 + \tilde{c}(x_i - 2^{\lambda_1})$, we recover

$$x_i = 2^{\lambda_1} + \tau_2 \,,$$

which concludes the proof. □

Corollary 2. *The* `JOIN` *protocol is zero-knowledge w.r.t. the group manager. Furthermore, the user's membership secret key x_i is a random integer from Λ.*

Proof. Straight-forward. □

Corollary 3. *In the random oracle model the group signature scheme presented in Section 4 is secure under the strong RSA and the decisional Diffie-Hellman assumption.*

Proof. We have to show that our scheme satisfies all the security properties listed in Definition 1.

Correctness: By inspection.

Unforgeability: Only group members are able to sign messages on behalf of the group: This is an immediate consequence of Theorem 2 and the random oracle model, that is, if we assume the hash function $\mathcal{H}$ behaves as a random function.

Anonymity: Given a valid signature $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$ identifying the actual signer is computationally hard for everyone but the group manager: Because of Theorem 2 the underlying interactive protocol is statistically zero-knowledge, no information is statistically revealed by (c, s_1, s_2, s_3, s_4) in the random oracle model. Deciding whether some group member with certificate $[A_i, e_i]$ originated requires deciding whether the three discrete logarithms $\log_y T_1/A_i$, $\log_g T_2$, and $\log_g T_3/g^{e_i}$ are equal. This is assumed to be infeasible under the decisional Diffie-Hellman assumption and hence anonymity is guaranteed.

Unlinkability: Deciding if two signatures $(T_1, T_2, T_3, c, s_1, s_2, s_3, s_4)$ and $(\tilde{T}_1, \tilde{T}_2, \tilde{T}_3, \tilde{c}, \tilde{s}_1, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4)$ were computed by the same group member is computationally hard. Simiarly as for Anonymity, the problem of linking two signatures reduces to decide whether the three discrete logarithms $\log_y T_1/\tilde{T}_i$, $\log_g T_2/\tilde{T}_2$, and $\log_g T_3/\tilde{T}_3$ are equal. This is, however, impossible under Decisional Diffie-Hellman Assumption.

Exculpability: Neither a group member nor the group manager can sign on behalf of other group members: First note that due to Corollary 2, GM does not get any information about a user's secret x_i apart from a^{x_i}. Thus, the value x_i is computationally hidden from GM. Next note that T_1, T_2, and T_3 are an unconditionally binding commitments to A_i and e_i. One can show that, if the factorization of n would be publicly known, the interactive proof underlying the group signature scheme is a proof of knowledge of the discrete log of $A_i^{e_i}/a_0$ (provided that ℓ_p is larger than twice to output length of the hash function / size of the challenges). Hence, not even the group manager can sign on behalf of P_i because computing discrete logarithms is assumed to be infeasible.

Traceability: The group manager is able to open any valid group signature and *provably* identify the actual signer: Assuming that the signature is valid, this implies that T_1 and T_2 are of the required form and so A_i can be uniquely recovered. Due to Theorem 1 a group certificate $[A_i = A(x_i), e_i]$ with $x_i \in \Lambda$

and $e_i \in \Gamma$ can only be obtained from via the `JOIN` protocol. Hence, the A_i recovered can be uniquely be linked to an instance of the `JOIN` protocol and thus the user P_i who originated the signature can be identified.

Coalition-resistance: Assuming the random oracle model, this follows from Theorems 1 and 2.

□

Corollary 4. *The identity escrow scheme derived from our group signature scheme is secure under the strong RSA and the decisional Diffie-Hellman assumption.*

Proof. The proof is essentially the same as for Corollary 3, the difference being that we do not need the random oracle model but can apply Theorems 1 and 2 directly. □

7 Conclusions

This paper presents a very efficient and provably secure group signature scheme and a companion identity escrow scheme that are based on the strong RSA assumption. Their performance and security appear to significantly surpass those of prior art. Extending the scheme to a blind group-signature scheme or to split the group manager into a membership manager and a revocation manager is straight-forward (cf. [CM98a, LR98]).

References

[ASW98] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *LNCS*, pages 591–606, Springer-Verlag, 1998.

[Ate99] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *6st ACM Conference on Computer and Communication Security*, ACM Press, 1999.

[BF97] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 480–494, Springer-Verlag, 1997.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communication Security*, pp. 62–73, ACM Press, 1993.

[Bon98] D. Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory (ANTS-III)*, vol. 1423 of *LNCS*, pp. 48–63, Springer-Verlag, 1998.

[Bra93] S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, Centrum voor Wiskunde en Informatica, April 1993.

[CD98] J. Camenisch and I. Damgård. Verifiable encryption and applications to group signatures and signature sharing. BRICS Technical Report, RS-98-32.

[CM98a] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology — ASIACRYPT '98*, vol. 1514 of *LNCS*, pp. 160–174, Springer-Verlag, 1998.

[CM98b] ———. A group signature scheme based on an RSA-variant. Technical Report RS-98-27, BRICS, University of Aarhus, November 1998. An earlier version appears in [CM98a].

[CM99a] ———. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology — EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 107–122, Springer-Verlag, 1999.

[CM99b] ———. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology — CRYPTO '99*, vol. 1666 of *LNCS*, pp. 413–430, Springer-Verlag, 1999.

[CP95] L. Chen and T. P. Pedersen. New group signature schemes. In *Advances in Cryptology — EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 171–181, 1995.

[CS97] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424, Springer-Verlag, 1997.

[Cam98] J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. PhD thesis, vol. 2 of *ETH Series in Information Security an Cryptography*, Hartung-Gorre Verlag, Konstanz, 1998. ISBN 3-89649-286-1.

[Cop96] D. Coppersmith. Finding a small root of a bivariatre interger equation; factoring with high bits known. In *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *LNCS*, pages 178–189. Springer Verlag, 1996.

[CvH91] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 257–265, Springer-Verlag, 1991.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6): 644–654, 1976.

[FO97] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — CRYPTO '97*, vol. 1297 of *LNCS*, pp. 16–30, Springer-Verlag, 1997.

[FO98] ———. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In *Advances in Cryptology — EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 32–46, Springer-Verlag, 1998.

[FS87] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194, Springer-Verlag, 1987.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[KP98] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology — CRYPTO '98*, vol. 1642 of *LNCS*, pp. 169–185, Springer-Verlag, 1998.

[LR98] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography (FC '98)*, vol. 1465 of *LNCS*, pp. 184–197, Springer-Verlag, 1998.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

GEM: a Generic Chosen-Ciphertext Secure Encryption Method

[Published in B. Preneel, Ed., *Topics in Cryptology – CT-RSA 2002*, vol. 2271 of *Lecture Notes in Computer Science*, pp. 263–276, Springer-Verlag, 2002.]

Jean-Sébastien Coron[1], Helena Handschuh[1], Marc Joye[2], Pascal Paillier[1], David Pointcheval[3], and Christophe Tymen[1,3]

[1] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{jean-sebastien.coron, helena.handschuh, pascal.paillier, christophe.tymen}@gemplus.com
[2] Gemplus Card International
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos Cedex, France
marc.joye@gemplus.com – http://www.geocities.com/MarcJoye/
[3] École Normale Supérieure, Computer Science Department
45 rue d'Ulm, 75230 Paris Cedex 05, France
david.pointcheval@ens.fr – http://www.di.ens.fr/~pointche/

Abstract. This paper proposes an efficient and provably secure transform to encrypt a message with any asymmetric one-way cryptosystem. The resulting scheme achieves adaptive chosen-ciphertext security in the random oracle model.
Compared to previous known generic constructions (Bellare, Rogaway, Fujisaki, Okamoto, and Pointcheval), our embedding reduces the encryption size and/or speeds up the decryption process. It applies to numerous cryptosystems, including (to name a few) ElGamal, RSA, Okamoto-Uchiyama and Paillier systems.

Keywords. Public-key encryption, hybrid encryption, chosen-ciphertext security, random oracle model, generic conversion, block ciphers, stream ciphers.

1 Introduction

A major contribution of cryptography is *information privacy*: through encryption, parties can securely exchange data over an insecure channel. Loosely speaking this means that unauthorized recipients can learn nothing useful about the exchanged data.

Designing a "good" encryption scheme is a very challenging task. There are basically two criteria to compare the performances of encryption schemes: *efficiency* and *security*. Security is measured as the ability to resist attacks in a

given adversarial model [1, 8]. The standard security notion is IND-CCA2 *security*, *i.e.*, indistinguishability under adaptive chosen-ciphertext attacks (cf. Section 2). Usually, an (asymmetric) encryption scheme is proven secure by exhibiting a *reduction*: if an adversary can break the IND-CCA2 security then the same adversary can solve a related problem assumed to be infeasible.

This paper is aimed at simplifying the security proof by providing a *Generic Encryption Method* (GEM) to convert *any* asymmetric one-way cryptosystem into a *provably secure* encryption scheme. Hence, when a new asymmetric one-way function is identified, one can easily design a secure encryption scheme. Moreover, the conversion we propose is very efficient (computationally and memory-wise): the converted scheme has roughly the same cost as that of the one-way cryptosystem it is built from.

1.1 Previous work

In [3], Bellare and Rogaway described OAEP, a generic conversion to transform a *"partial-domain" one-way trapdoor permutation* into an IND-CCA2 secure encryption scheme in the random oracle model [2, 7]. Later, Fujisaki and Okamoto [5] presented a way to transform, in the random oracle model, any *chosen-plaintext* (IND-CPA) secure encryption scheme into an IND-CCA2 one. They improved their results in [6] where they gave a generic method to convert a *one-way* (OW-CPA) cryptosystem into an IND-CCA2 secure encryption scheme in the random oracle model. A similar result was independently discovered by Pointcheval [13]. More recently, Okamoto and Pointcheval [12] proposed a more efficient generic conversion, called REACT, to convert any one-way cryptosystem secure under *plaintext-checking attacks* (OW-PCA) into an IND-CCA2 encryption scheme. Contrary to [5, 6, 13], re-encryption is unnecessary in the decryption process to ensure IND-CCA2 security.

1.2 Our results

This paper presents GEM, a generic IND-CCA2 conversion. The converted scheme, $\mathbb{E}_{pk}$, built from any OW-PCA asymmetric encryption $\mathcal{E}_{pk}$ and any length-preserving IND-secure symmetric encryption scheme $\mathtt{E}_K$, is secure in the sense of IND-CCA2, in the random oracle model. As discussed in Section 2, the security levels we require for $\mathcal{E}_{pk}$ and $\mathtt{E}_K$ are *very* weak and the security level we obtain for $\mathbb{E}_{pk}$ is very high.

1.3 Organization of the paper

The rest of this paper is organized as follows. In Section 2, we review the security notions for encryption, in both the symmetric and the asymmetric settings. Section 3 is the core of the paper. We present our new padding to convert, in the random oracle model, any asymmetric one-way cryptosystem into an encryption scheme that is secure in the *strongest* sense. We prove the security of our construction in Section 4 by providing and proving a *concrete* reduction algorithm. Finally, we illustrate the merits of our conversion in Section 5.

2 Security Notions

In this section, we recall the definition of an encryption scheme and discuss some related security notions. A good reference to the subject is [1].

2.1 Encryption schemes

Definition 1. *An* encryption scheme *consists of three algorithms* $(\mathcal{K}, \mathcal{E}_{pk}, \mathcal{D}_{sk})$.

1. *On input a security parameter k, the key generation algorithm $\mathcal{K}(1^k)$ outputs a random matching pair (pk, sk) of encryption/decryption keys. For the symmetric case, we assume wlog that the encryption and decryption keys are identical, $K := pk = sk$. The key pk is public and the keys sk and K are secret.*
2. *The encryption algorithm $\mathcal{E}_{pk}(m, r)$ outputs a ciphertext c corresponding to a plaintext $m \in \mathtt{MSPC} \subseteq \{0,1\}^*$, using the random coins $r \in \Omega$. When the process is deterministic, we simply write $\mathcal{E}_{pk}(m)$. In the symmetric case, we note $\mathtt{E}_K$ instead of $\mathcal{E}_{pk}$.*
3. *The decryption algorithm $\mathcal{D}_{sk}(c)$ outputs the plaintext m associated to the ciphertext c or a notification $\perp$ that c is not a valid ciphertext. In the symmetric case, we use the notations $\mathtt{D}_K$.*

Furthermore, we require that for all $m \in \mathtt{MSPC}$ and $r \in \Omega$, $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, r)) = m$.

The converted scheme we propose in this paper is a combination of an asymmetric encryption scheme and a length-preserving symmetric encryption scheme. We assume very weak security properties from those two schemes. Namely, we require that the asymmetric scheme is OW-PCA and that the symmetric scheme is IND, as defined below.

2.2 Security requirements

An attacker is said *passive* if, in addition to the ciphertext, s/he only obtains some auxiliary information s, which may depend on the potential plaintexts (but not on the key) [9, § 1.5] and other public information (*e.g.*, the security parameter k and the public key pk). Note that in the asymmetric case an attacker can always construct valid pairs of plaintext/ciphertext from the public encryption key pk.

A minimal security requirement for an encryption scheme is *one-wayness* (OW). This captures the property that an adversary cannot recover the *whole* plaintext from a given ciphertext. In some cases, partial information about a plaintext may have disastrous consequences. This is captured by the notion of *semantic security* or the equivalent notion of *indistinguishability* [10]. Basically, indistinguishability means that the only strategy for an adversary to distinguish the encryptions of any two plaintexts is by guessing at random.

In the asymmetric case, suppose that the attacker has access to an oracle telling whether a pair (m, c) of plaintext/ciphertext is valid; *i.e.*, whether

$m = \mathcal{D}_{sk}(c)$ holds. Following [12], such an attack scenario is referred to as the *plaintext-checking attack* (PCA). From the pair of adversarial goal (OW) and adversarial model (PCA), we derive the security notion of OW-PCA.

Definition 2. *An asymmetric encryption scheme is* OW-PCA *if no attacker with access to a plaintext-checking oracle* $\mathcal{O}^{\text{PCA}}$ *can recover the whole plaintext corresponding to a ciphertext with non-negligible probability. More formally, an asymmetric encryption scheme is* (τ, q, ε)*-secure in the sense of* OW-PCA *if for any adversary* $\mathcal{A}$ *which runs in time at most* τ*, makes at most* q *queries to* $\mathcal{O}^{\text{PCA}}$*, its success* ε *satisfies*

$$\Pr_{\substack{m \leftarrow \{0,1\}^* \\ r \leftarrow \Omega}} \left[\begin{array}{c} (sk, pk) \leftarrow \mathcal{K}(1^k), c \leftarrow \mathcal{E}_{pk}(m, r) : \\ \mathcal{A}^{\mathcal{O}^{\text{PCA}}}(c, s) = m \end{array} \right] \leq \varepsilon$$

where the probability is also taken over the random choices of $\mathcal{A}$*.*

For the symmetric case, we consider a passive attacker who tries to break the indistinguishability property of the encryption scheme. The attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ runs in two stages. In the first stage (or *find stage*), on input k, $\mathcal{A}_1$ outputs a pair of messages (m_0, m_1) and some auxiliary information s. Next, in the second stage (or *guess stage*), given the encryption c_b of either m_0 or m_1 and the auxiliary information s, $\mathcal{A}_2$ tells if the challenge ciphertext c_b encrypts m_0 or m_1.

Definition 3. *A symmetric encryption scheme is* IND *if no attacker can distinguish the encryptions of two equal-length plaintexts with probability non-negligibly greater than* $1/2$*. More formally, a symmetric encryption scheme is* (τ, ν)*-secure in the sense of* IND *if for any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *which runs in time at most* τ*, its advantage* ν *satisfies*

$$\Pr_{b \leftarrow \{0,1\}} \left[\begin{array}{c} K \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow \mathcal{A}_1(k), \\ c_b \leftarrow \mathtt{E}_K(m_b) : \mathcal{A}_2(m_0, m_1, c_b, s) = b \end{array} \right] \leq \frac{1+\nu}{2}$$

where the probability is also taken over the random choices of $\mathcal{A}$*.*

In contrast, for our converted encryption scheme we require the *highest* security level, namely IND-CCA2 security. The notion of IND-CCA2 for an asymmetric encryption scheme considers an *active* attacker who tries to break the system by probing with chosen-ciphertext messages. Such an attack can be non-adaptive (CCA1) [11] or adaptive (CCA2) [14]. In a CCA2 scenario, the adversary may run a second chosen ciphertext attack upon receiving the challenge ciphertext c_b (the only restriction being not to probe on c_b).

Definition 4. *An asymmetric encryption scheme is* IND-CCA2 *if no attacker with access to a decryption oracle* $\mathcal{O}^{\mathcal{D}_{sk}}$ *can distinguish the encryptions of two equal-length plaintexts with probability non-negligibly greater than* $1/2$*. More formally, an asymmetric encryption scheme is* (τ, q, ε)*-secure in the sense of* IND-CCA2 *if for any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *which runs in time at most* τ*, makes at*

most q queries to $\mathcal{O}^{\mathcal{D}_{sk}}$, its advantage ε satisfies

$$\Pr_{\substack{b\leftarrow\{0,1\}\\ r\leftarrow\Omega}}\left[\begin{array}{c}(sk,pk)\leftarrow\mathcal{K}(1^k),\\ (m_0,m_1,s)\leftarrow{\mathcal{A}_1}^{\mathcal{O}^{\mathcal{D}_{sk}}}(k,pk),\\ c_b\leftarrow\mathcal{E}_{pk}(m_b,r):\\ {\mathcal{A}_2}^{\mathcal{O}^{\mathcal{D}_{sk}}}(m_0,m_1,c_b,s)=b\end{array}\right]\leq\frac{1+\varepsilon}{2}$$

where the probability is also taken over the random choices of $\mathcal{A}$, and $\mathcal{A}_2$ is not allowed to query on c_b.

3 GEM: Generic Encryption Method

A very appealing way to encrypt a message consists in using a *hybrid encryption* mode. A random session key R is first encrypted with an asymmetric cryptosystem. Then the message is encrypted under that session key with a symmetric cryptosystem. Although seemingly sound, this scheme does not achieve IND-CCA2 security under weak security assumptions for the two underlying cryptosystems. This section shows how to modify the above paradigm for attaining the IND-CCA2 security level.

3.1 REACT transform

The authors of REACT imagined to append a checksum to the previous construction and prove the IND-CCA2 security of the resulting scheme in the random oracle model [12]. Briefly, REACT works as follows. A plaintext m is transformed into the ciphertext (c_1, c_2, c_3) given by

$$\text{REACT}(m)=\underbrace{\mathcal{E}_{pk}(R,u)}_{=c_1}\,\|\,\underbrace{\mathtt{E}_K(m)}_{=c_2}\,\|\,\underbrace{\mathrm{H}(R,m,c_1,c_2)}_{=c_3}$$

where u is a random, $K=\mathrm{G}(R)$, and G,H are hash functions.

Building on this, we propose a new generic encryption method. Our method is aimed at shortening the whole ciphertext by incorporating the checksum (*i.e.*, c_3) into c_1 while maintaining the IND-CCA2 security level, in the random oracle model.

3.2 New method

Let $\mathcal{E}_{pk}$ and $\mathtt{E}_K$ denote an asymmetric and a length-preserving symmetric encryption algorithms, respectively, and let $\mathrm{F},\mathrm{G},\mathrm{H}$ denote hash functions. Let also $\mathcal{D}_{sk}$ and $\mathtt{D}_K$ denote the decryption algorithms corresponding to $\mathcal{E}_{pk}$ and $\mathtt{E}_K$, respectively. For convenience, for any element x defined over a domain A, we write $\sharp\, x$ for $|\{x\in A\}|$. So, for example, $\sharp\, m$ represents the cardinality of the message space, *i.e.*, the number of different plaintexts.

Encryption

Input: Plaintext m, random $\rho = r \| u$.
Output: Ciphertext (c_1, c_2) given by

$$\mathbb{E}_{pk}(m, \rho) = \underbrace{\mathcal{E}_{pk}(w, u)}_{=c_1} \,\|\, \underbrace{\mathtt{E}_K(m)}_{=c_2}$$

where $s = \mathrm{F}(m, r)$, $w = s \,\|\, r \oplus \mathrm{H}(s)$,
and $K = \mathrm{G}(w, c_1)$.

Decryption

Input: Ciphertext (c_1, c_2).
Output: Plaintext $\dot{m}$ or symbol $\perp$ according to

$$\mathbb{D}_{sk}(c_1 \| c_2) = \begin{cases} \dot{m} & \text{if } \dot{s} = \mathrm{F}(\dot{m}, \dot{r}) \\ \perp & \text{otherwise} \end{cases}$$

where $\dot{w} := \dot{s} \| \dot{t} = \mathcal{D}_{sk}(c_1)$, $\dot{K} = \mathrm{G}(\dot{w}, c_1)$,
$\dot{m} = \mathtt{D}_{\dot{K}}(c_2)$, and $\dot{r} = \dot{t} \oplus \mathrm{H}(\dot{s})$.

4 Security Analysis

We now prove the security of our conversion. We show that if the hybrid encryption scheme $\mathbb{E}_{pk}$ can be broken under an adaptive chosen-ciphertext attack then either the length-preserving symmetric encryption scheme $\mathtt{E}_K$ or the asymmetric encryption scheme $\mathcal{E}_{pk}$ underlying our construction is *highly* insecure, namely the IND-security of $\mathtt{E}_K$ or the OW-PCA security of $\mathcal{E}_{pk}$ gets broken.

Theorem 1. *Suppose that there exists an adversary that breaks, in the random oracle model, the* IND-CCA2 *security of our converted scheme* $\mathbb{E}_{pk}$ *within a time bound* τ*, after at most* $q_{\mathrm{F}}, q_{\mathrm{G}}, q_{\mathrm{H}}, q_{\mathbb{D}_{sk}}$ *queries to hash functions* $\mathrm{F}, \mathrm{G}, \mathrm{H}$ *and decryption oracle* $\mathcal{O}^{\mathbb{D}_{sk}}$*, respectively, and with an advantage* ε*. Then, for all* $0 < \nu < \varepsilon$*, there exists*

- *an adversary that breaks the* IND *security of* $\mathtt{E}_K$ *within a time bound* τ *and an advantage* ν*; or*
- *an adversary with access to a plaintext-checking oracle* $\mathcal{O}^{\mathrm{PCA}}$ *(responding in time bounded by* τ_{PCA}*) that breaks the* OW-PCA *security of* $\mathcal{E}_{pk}$ *within a time bound*

$$\tau' = \tau + (q_{\mathrm{F}}\, q_{\mathrm{H}} + q_{\mathrm{G}} + q_{\mathbb{D}_{sk}}(q_{\mathrm{F}} + q_{\mathrm{G}}))\,(\tau_{\mathrm{PCA}} + O(1))\,,$$

after at most

$$q_{\mathrm{PCA}} \leq q_{\mathrm{F}}\, q_{\mathrm{H}} + q_{\mathrm{G}} + q_{\mathbb{D}_{sk}}(q_{\mathrm{F}} + q_{\mathrm{G}})$$

queries to $\mathcal{O}^{\mathrm{PCA}}$*, and with success probability*

$$\varepsilon' \geq \frac{\varepsilon - \nu}{2} - \frac{q_{\mathrm{F}}}{\sharp\, r} - q_{\mathbb{D}_{sk}} \left(\tfrac{1}{\sharp\, s} + q_{\mathrm{F}} \left(\tfrac{1}{\sharp\, r} + \tfrac{1}{\sharp\, s} \right) + \nu + \tfrac{1}{\sharp\, m} \right) \quad .$$

From this, we immediately obtain:

Corollary 1. *For any* OW-PCA *asymmetric encryption* $\mathcal{E}_{pk}$ *and any length-preserving* IND*-secure symmetric encryption scheme* $\mathtt{E}_K$*, our converted scheme* $\mathbb{E}_{pk}$ *is* IND-CCA2 *secure in the random oracle model.* □

To prove Theorem 1, we suppose that there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ able to break the INC-CCA2 security of $\mathbb{E}_{pk}$. We further suppose that $\mathtt{E}_K$ is (τ, ν)-secure in the sense of IND. From $\mathcal{A}$, we then exhibit an adversary $\mathcal{B}$ (*i.e.*, a reduction algorithm) that inverts $\mathcal{E}_{pk}$ using a plaintext-checking oracle, and thus breaks the OW-PCA security of $\mathcal{E}_{pk}$.

4.1 A useful lemma

The assumption that $\mathtt{E}_K$ is length-preserving and (τ, ν)-IND secure implies the following lemma.

Lemma 1. *Assume that* $\mathtt{E}_K$ *is a length-preserving* (τ, ν)-IND *symmetric encryption scheme, where* τ *denotes the time needed for evaluating* $\mathtt{E}_K(\cdot)$*. Then given a pair* (m, c) *of plaintext/ciphertext, we have*

$$\Pr_K[\mathtt{E}_K(m) = c] \leq \nu + \frac{1}{\sharp m} \ .$$

Proof. Given the pair (m, c), we consider the following distinguisher $\mathcal{A}$. $\mathcal{A}$ randomly chooses a bit $d \in \{0, 1\}$, sets $m_d = m$ and $m_{\neg d}$ to a random value m'. The pair $(m_d, m_{\neg d})$ is then sent to the encryption oracle which returns $c_b = \mathtt{E}_K(m_b)$ for a random key K and a random $b \in \{0, 1\}$. $\mathcal{A}$ then checks if $c_b = c$, returns d if the equality holds and $\neg d$ otherwise. Letting ε the advantage of $\mathcal{A}$, we have

$$\begin{aligned}
\varepsilon &= 2 \Pr_{m',K,d,b}[\mathcal{A} \text{ returns } b] - 1 \\
&= 2 \Pr_{m',K,d,b}[(c_b = c) \wedge (d = b)] + \\
&\qquad 2 \Pr_{m',K,d,b}[(c_b \neq c) \wedge (\neg d = b)] - 1 \\
&= 2 \Pr_{m',K,d,b}[(\mathtt{E}_K(m) = c) \wedge (d = b)] + \\
&\qquad 2 \Pr_{m',K,d,b}[(\mathtt{E}_K(m') \neq c) \wedge (\neg d = b)] - 1 \\
&= \Pr_K[\mathtt{E}_K(m) = c] + \Pr_{m',K}[\mathtt{E}_K(m') \neq c] - 1 \\
&= \Pr_K[\mathtt{E}_K(m) = c] - \Pr_{m',K}[\mathtt{E}_K(m') = c] \leq \nu \ .
\end{aligned}$$

The proof follows by noting that as $\mathtt{E}_K$ is length-preserving, it permutes the set of messages for any key K and so $\Pr_{m',K}[\mathtt{E}_K(m') = c] = 1/\sharp m$. □

4.2 Description of the reduction algorithm

$\mathcal{B}$ is given a challenge encryption $y = \mathcal{E}_{pk}(\widetilde{w}, *)$, an oracle $\mathcal{O}^{\text{PCA}}$ which answers plaintext-checking requests on $\mathcal{E}_{pk}$, and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the IND-CCA2 security of $\mathbb{E}_{pk}$. $\mathcal{B}$'s goal is to retrieve all the bits of $\widetilde{w}$. Wlog, we assume that $\mathcal{O}^{\text{PCA}}$ responds to any of $\mathcal{B}$'s requests with no error and within a time bounded by τ_{PCA}.

Throughout, the following notations are used. For any predicate $\text{R}(x)$, $\text{R}(*)$ stands for $\exists x$ s.t. $\text{R}(x)$. If $\mathcal{O}$ is an oracle to which $\mathcal{A}$ has access, we denote by *query* $\mapsto$ *response* the correspondence $\mathcal{O}$ establishes between $\mathcal{A}$'s request *query* and the value *response* returned to $\mathcal{A}$. $\textsc{Hist}\,[\mathcal{O}]$ stands for the set of correspondences established by $\mathcal{O}$ as time goes on: $\textsc{Hist}\,[\mathcal{O}]$ can be seen as an history tape which gets updated each time $\mathcal{A}$ makes a query to $\mathcal{O}$. We denote by $q_{\mathcal{O}}$ the number of calls $\mathcal{A}$ made to $\mathcal{O}$ during the simulation.

Overview of $\mathcal{B}$ At the beginning, $\mathcal{B}$ chooses a random value $\widetilde{K}$. $\mathcal{B}$ then runs $\mathcal{A}$ and provides a simulation for F, G, H and $\mathbb{D}_{sk}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ runs in two stages. At the end of the first stage (find stage), $\mathcal{A}_1$ outputs a pair (m_0, m_1). $\mathcal{B}$ then randomly chooses $b \in \{0, 1\}$, computes $\widetilde{c_2} = \mathtt{E}_{\widetilde{K}}(m_b)$ and builds $(y, \widetilde{c_2})$. This challenge is provided to $\mathcal{A}_2$, which outputs some bit at the end of the second stage (guess stage). Once finished, $\mathcal{B}$ checks whether some $\widetilde{w}$ has been defined during the game. If so, $\widetilde{w}$ is returned as the inverse of $\mathcal{E}_{pk}$ on y; otherwise a failure answer is returned. The detailed description of the simulation follows.

Wlog, we assume that $\mathcal{A}$ keeps track of all the queries throughout the game so that $\mathcal{A}$ never has to make the same query twice to the same oracle.

Simulation of F For each new query (m, r),

- (Event E_1) if processing guess stage and $m = m_b$ and there exists $s \mapsto h \in \textsc{Hist}\,[\text{H}]$ such that $y = \mathcal{E}_{pk}(s \| r \oplus h, *)$ then F sets $\widetilde{w} := s \| r \oplus h$, returns s and updates its history,
- (no event) else F outputs a random value and updates its history.

Simulation of G For each new query (w, c_1),

- (Event E_2) if $c_1 = y$ and $y = \mathcal{E}_{pk}(w, *)$ then G sets $\widetilde{w} := w$, returns $\widetilde{K}$ and updates its history,
- (Event E_3) else if $c_1 \neq y$ and $y = \mathcal{E}_{pk}(w, *)$ then G sets $\widetilde{w} := w$, returns a random value and updates its history,
- (no event) else G outputs a random value and updates its history.

Simulation of H For each new query s, H outputs a random value and updates its history.

Simulation of $\mathbb{D}_{sk}$ (plaintext extractor) For each new query (c_1, c_2), $\mathbb{D}_{sk}$ first checks (this verification step only stands while the guess stage $\mathcal{A}_2$ is running) that $(c_1, c_2) \neq (y, \widetilde{c_2})$ since if this equality holds, the query must be rejected as $\mathcal{A}$ attempts to decrypt its own challenge ciphertext. Then, $\mathbb{D}_{sk}$ tries to find the only (if any) message m matching the query. To achieve this, $\mathbb{D}_{sk}$ invokes the simulation of G, H and F provided by $\mathcal{B}$ as follows.

- Find the unique pair (r, s) such that $(*, r) \mapsto s \in \textsc{Hist}\,[\mathrm{F}]$, $s \mapsto h \in \textsc{Hist}\,[\mathrm{H}]$ and $c_1 = \mathcal{E}_{pk}(s \| r \oplus h, *)$. If such a pair exists,
  - query G to get $K = \mathrm{G}(s \| r \oplus h, c_1)$,
  - letting $m = \mathtt{D}_K(c_2)$, query F to check if $\mathrm{F}(m, r) = s$. If the equality holds, return m; otherwise reject the query (Event RJ_1).
- If the search for (r, s) is unsuccessful, check if there exists w with $(w, c_1) \mapsto K \in \textsc{Hist}\,[\mathrm{G}]$ and $c_1 = \mathcal{E}_{pk}(w, *)$. If such an w exists,
  - define s and t by $w = s \| t$, and query H to get $h = \mathrm{H}(s)$,
  - letting $m = \mathtt{D}_K(c_2)$, query F to check if $\mathrm{F}(m, t \oplus h) = s$. If the equality holds, return m; otherwise reject the query (Event RJ_2).
- If the search for w is unsuccessful, reject the query (Event RJ_3).

4.3 Soundness of $\mathcal{B}$

Unless otherwise mentioned, all probabilities are taken over the random choices of $\mathcal{A}$ and $\mathcal{B}$.

Simulation of random oracles The plaintext $\widetilde{w}$ uniquely defines $\widetilde{s}$ and $\widetilde{t}$ such that $\widetilde{w} = \widetilde{s} \| \widetilde{t}$. We note $\widetilde{r}$ the random variable $\widetilde{t} \oplus \mathrm{H}(\widetilde{s})$.

Soundness of F. The simulation of F fails when $(m_b, \widetilde{r})$ is queried and answered with some value $s \neq \widetilde{s}$ before $\widetilde{s}$ appears in $\textsc{Hist}\,[\mathrm{H}]$.

Let q_{F}^1 denote the number of oracle queries $\mathcal{A}_1$ made to F during the find stage. Since $\tilde{r}$ is a uniformly-distributed random variable throughout the find stage, we certainly have $\Pr[\mathrm{F}\ \mathsf{incorrect\ in\ the\ find\ stage}] \leq q_{\mathrm{F}}^1 / \sharp\, r$.

Moreover, throughout the guess stage, $\mathcal{A}_2$ cannot gain any information about $\widetilde{r}$ without knowing $\mathrm{H}(\widetilde{s})$ because H is a random function. Hence, letting q_{F}^2 the number of oracle queries $\mathcal{A}_2$ made to F during the guess stage, we have $\Pr[\mathrm{F}\ \mathsf{incorrect\ in\ the\ guess\ stage}] \leq q_{\mathrm{F}}^2 / \sharp\, r$.

Consequently, the probability that an error occurs while $\mathcal{B}$ simulates the oracle F is upper-bounded by

$$\Pr[\mathrm{F}\ \mathsf{incorrect}] \leq \frac{q_{\mathrm{F}}^1 + q_{\mathrm{F}}^2}{\sharp\, r} = \frac{q_{\mathrm{F}}}{\sharp\, r} \ .$$

Soundness of G. The simulation is perfect.

Soundness of H. The simulation is perfect.

Plaintext extraction The simulation of $\mathbb{D}_{sk}$ fails when $\mathbb{D}_{sk}$ returns $\perp$ although the query $c = (c_1, c_2)$ is a valid ciphertext. Let then m, r, s, t, h, w, K be the unique random variables associated to c in this case. Obviously, c was rejected through event RJ_3, because a rejection through RJ_1 or RJ_2 refutes the validity of c. Therefore, if $\mathbb{D}_{sk}$ is incorrect for c, we must have

$$\Big(\underbrace{(m,r) \notin \textsc{Hist}\,[\mathrm{F}]}_{:=\neg\mathsf{E}_\mathrm{F}} \vee \underbrace{s \notin \textsc{Hist}\,[\mathrm{H}]}_{:=\neg\mathsf{E}_\mathrm{H}}\Big) \wedge \underbrace{\big((w,c_1) \mapsto K \notin \textsc{Hist}\,[\mathrm{G}]\big)}_{:=\neg\mathsf{E}_\mathrm{G}} \ .$$

Hence,

$$\begin{aligned}
\Pr[\mathbb{D}_{sk} \text{ incorrect for } c] &= \Pr[(\neg\mathsf{E}_\mathrm{F} \vee \neg\mathsf{E}_\mathrm{H}) \wedge \neg\mathsf{E}_\mathrm{G}] \\
&= \Pr[((m,r) \neq (m_b,\widetilde{r})) \wedge (\neg\mathsf{E}_\mathrm{F} \vee \neg\mathsf{E}_\mathrm{H}) \wedge \neg\mathsf{E}_\mathrm{G}] + \\
&\quad \Pr[((m,r) = (m_b,\widetilde{r})) \wedge (\neg\mathsf{E}_\mathrm{F} \vee \neg\mathsf{E}_\mathrm{H}) \wedge \neg\mathsf{E}_\mathrm{G}] \\
&\leq \Pr[((m,r) \neq (m_b,\widetilde{r})) \wedge (\neg\mathsf{E}_\mathrm{F} \vee \neg\mathsf{E}_\mathrm{H})] + \Pr[((m,r) = (m_b,\widetilde{r})) \wedge \neg\mathsf{E}_\mathrm{G}] \\
&= \Pr[((m,r) \neq (m_b,\widetilde{r})) \wedge \neg\mathsf{E}_\mathrm{F}] + \Pr[((m,r) \neq (m_b,\widetilde{r})) \wedge (\mathsf{E}_\mathrm{F} \wedge \neg\mathsf{E}_\mathrm{H})] + \\
&\quad \Pr[((m,r) = (m_b,\widetilde{r})) \wedge \neg\mathsf{E}_\mathrm{G}] \ .
\end{aligned}$$

1. Assume $(m,r) \neq (m_b,\widetilde{r})$ and $(m,r) \notin \textsc{Hist}\,[\mathrm{F}]$. Since F is a random function, $\mathrm{F}(m,r)$ is a uniformly distributed random value unknown to $\mathcal{A}$. The fact that c is a valid ciphertext implies that $\mathrm{F}(m,r) = s$, which happens with probability

$$\Pr[c \text{ is valid} \wedge (m,r) \notin \textsc{Hist}\,[\mathrm{F}]] = \frac{1}{\sharp\, s} \ .$$

2. Assume $(m,r) \neq (m_b,\widetilde{r})$ and $(m,r) \in \textsc{Hist}\,[\mathrm{F}] \wedge s \mapsto h \notin \textsc{Hist}\,[\mathrm{H}]$. Suppose that $s \neq \widetilde{s}$. Since H is a random function, $\mathrm{H}(s)$ is a uniformly distributed random value unknown to $\mathcal{A}$. The validity of c implies that $(m, t \oplus \mathrm{H}(s)) \mapsto s \in \textsc{Hist}\,[\mathrm{F}]$, which happens with probability

$$\Pr[(m, t \oplus \mathrm{H}(s)) \mapsto s \in \textsc{Hist}\,[\mathrm{F}]] \leq \frac{q_\mathrm{F}}{\sharp\, r} \ .$$

   Now assume $s = \widetilde{s}$. In this case, we must have $(m,r) \mapsto \widetilde{s} \in \mathrm{F}$ which occurs with probability

$$\Pr[(m,r) \mapsto \widetilde{s} \in \textsc{Hist}\,[\mathrm{F}]] \leq \frac{q_\mathrm{F}}{\sharp\, s} \ .$$

3. Assume $(m,r) = (m_b,\widetilde{r})$ and $(w,c_1) \mapsto K \notin \textsc{Hist}\,[\mathrm{G}]$. This implies $s = \widetilde{s}$, $t = \widetilde{t}$, $w = \widetilde{w}$ and $c_1 \neq y$. Hence, if c is valid, we must have $\mathtt{E}_K(m_b) = c_2$ for a uniformly distributed K. By virtue of Lemma 1, this is bounded by

$$\Pr_K[\mathtt{E}_K(m_b) = c_2] \leq \nu + \frac{1}{\sharp\, m} \ .$$

   Gathering all preceding bounds, we get

$$\begin{aligned}
&\Pr[c \text{ is valid} \wedge \mathbb{D}_{sk} \text{ incorrect for } c] \\
&\quad \leq \frac{1}{\sharp\, s} + q_\mathrm{F}\left(\frac{1}{\sharp\, r} + \frac{1}{\sharp\, s}\right) + \nu + \frac{1}{\sharp\, m} \,,
\end{aligned}$$

which, taken over all queries of $\mathcal{A}$, leads to

$$\Pr[\mathbb{D}_{sk} \text{ incorrect}] \leq q_{\mathbb{D}_{sk}} \left(\frac{1}{\sharp s} + q_{\mathrm{F}} \left(\frac{1}{\sharp r} + \frac{1}{\sharp s} \right) + \nu + \frac{1}{\sharp m} \right) \ .$$

Conclusion We have

$$\begin{aligned} \Pr[\mathcal{B} \text{ incorrect}] &= \Pr[\mathrm{F} \text{ incorrect}] + \Pr[\mathbb{D}_{sk} \text{ incorrect}] \\ &\leq \frac{q_{\mathrm{F}}}{\sharp r} + q_{\mathbb{D}_{sk}} \left(\frac{1}{\sharp s} + q_{\mathrm{F}} \left(\frac{1}{\sharp r} + \frac{1}{\sharp s} \right) + \nu + \frac{1}{\sharp m} \right) \ . \end{aligned}$$

4.4 Reduction cost

Success probability Let us suppose that $\mathcal{A}$ distinguishes $\mathbb{E}_{pk}$ within a time bound τ with advantage ε in less than q_{F}, q_{H}, q_{G}, $q_{\mathbb{D}_{sk}}$ oracle calls. Defining $\Pr_2[\cdot] = \Pr[\cdot \mid \neg(\mathcal{B} \text{ incorrect})]$, this means that

$$\Pr_2[\mathcal{A} \to b] \geq \frac{1+\varepsilon}{2} \ .$$

Suppose also that $\mathtt{E}_K$ is (τ, ν)-indistinguishable. Assuming that the oracles are correctly simulated, if none of the events E_1, E_2 or E_3 occurs, then $\mathcal{A}$ never asked $\widetilde{r}$ to the random oracle F, neither did it learn the key $\widetilde{K}$ under which m_b was encrypted in $\widetilde{c}_2$ (this is due to the randomness of F and G). This upper-limits the information leakage on b by ν, since $\mathcal{A}$'s running time is bounded by τ. Noting $\mathsf{E}_{win} = \mathsf{E}_1 \vee \mathsf{E}_2 \vee \mathsf{E}_3$, this implies

$$\Pr_2[\mathcal{A} \to b \mid \neg \mathsf{E}_{win}] \leq \frac{1+\nu}{2} \ .$$

We then get

$$\begin{aligned} \frac{1+\varepsilon}{2} \leq \Pr_2[\mathcal{A} \to b] &\leq \Pr_2[\mathcal{A} \to b \mid \neg \mathsf{E}_{win}] + \Pr_2[\mathsf{E}_{win}] \\ &\leq \frac{1+\nu}{2} + \Pr_2[\mathsf{E}_{win}] \, , \end{aligned}$$

whence $\Pr_2[\mathsf{E}_{win}] \geq (\varepsilon - \nu)/2$. But $\Pr_2[\mathcal{B} \to \widetilde{w}] = \Pr_2[\mathsf{E}_{win}]$ and finally,

$$\begin{aligned} \Pr[\mathcal{B} \to \widetilde{w}] &\geq \Pr_2[\mathcal{B} \to \widetilde{w}] - \Pr[\mathcal{B} \text{ incorrect}] \\ &\geq \frac{\varepsilon - \nu}{2} - \frac{q_{\mathrm{F}}}{\sharp r} - \\ &\qquad q_{\mathbb{D}_{sk}} \left(\frac{1}{\sharp s} + q_{\mathrm{F}} \left(\frac{1}{\sharp r} + \frac{1}{\sharp s} \right) + \nu + \frac{1}{\sharp m} \right) \ . \end{aligned}$$

Hence $\mathcal{B}$ succeeds with non-negligible probability.

Total number of calls to $\mathcal{O}^{\text{PCA}}$ Checking that a pair of the form (w, y) satisfies $y = \mathcal{E}_{pk}(w, *)$ is done thanks to the plaintext-checking oracle $\mathcal{O}^{\text{PCA}}$. Therefore, oracle F makes at most $q_{\text{F}} \cdot q_{\text{H}}$ queries to $\mathcal{O}^{\text{PCA}}$, and oracle G makes at most $q_{\text{G}} \cdot 1$ queries to $\mathcal{O}^{\text{PCA}}$. Moreover, it is easy to see that oracle $\mathcal{O}^{\mathbb{D}_{sk}}$ makes at most $q_{\mathbb{D}_{sk}}(q_{\text{F}} + q_{\text{G}})$ calls since in the worst case $\mathbb{D}_{sk}$ has to call $\mathcal{O}^{\text{PCA}}$ for all elements $((*, r) \mapsto s) \in \text{HIST}\,[\text{F}]$ and for all elements $((w, c_1) \mapsto K) \in \text{HIST}\,[\text{G}]$. In conclusion, the total number of calls actually needed by $\mathcal{B}$ is upper-bounded by

$$q_{\text{PCA}} \leq q_{\text{F}}\, q_{\text{H}} + q_{\text{G}} + q_{\mathbb{D}_{sk}}(q_{\text{F}} + q_{\text{G}}) \ .$$

Total running time The reduction algorithm runs in time bounded by

$$\tau_{\mathcal{B}} = \tau + (q_{\text{F}}\, q_{\text{H}} + q_{\text{G}} + q_{\mathbb{D}_{sk}}(q_{\text{F}} + q_{\text{G}}))\,(\tau_{\text{PCA}} + O(1)) \ .$$

This completes the proof of Theorem 1.

5 Concluding Remarks

A very popular way to (symmetrically) encrypt a plaintext is to use a *stream cipher*. The simplest example is the Vernam cipher where a plaintext m is processed bit-by-bit to form the ciphertext c under the secret key K. With this cipher, each plaintext bit m_i is XOR-ed with the key bit K_i to produce the ciphertext bit $c_i = m_i \oplus K_i$. If K is truly random and changes for each plaintext m being encrypted, then the system is unconditionally secure. This ideal situation is, however, impractical for a real-world implementation. To resolve the key management problem, a stream cipher is usually combined with a public-key cryptosystem. Contrary to the obvious solution consisting in encrypting a random session key with an asymmetric cryptosystem and then using that key with a stream cipher, our hybrid scheme achieves provable security. Compared to a purely asymmetric solution, our scheme presents the advantage to encrypt *long* messages *orders of magnitude faster* thanks to the use of a symmetric cryptosystem.

Another merit of our scheme resides in its generic nature. The set of possible applications of the new conversion scheme is similar to that of REACT: it concerns *any* asymmetric function that is OW-PCA under a conjectured intractability assumption. A specificity of REACT is that it can operate "on the fly". The session key does not depend on the plaintext to be encrypted and can thus be computed in advance. This property is particularly advantageous when the asymmetric encryption is expensive, as is the case for discrete-log based cryptosystems. Remark that our scheme does not allow "on-the-fly" encryption, that was the price to pay for shortening the ciphertext.

In other cases such as for RSA (with a low encryption exponent, *e.g.*, 3 or $2^{16}+1$) or Rabin cryptosystem, on-the-fly encryption is not an issue and our scheme may be preferred because the resulting ciphertext is shorter. Furthermore, it is worth noting that for a deterministic asymmetric encryption scheme $\mathcal{E}_{pk}$, the

notions of OW-PCA and OW-CPA are identical: the validity of a pair (m, c) of plaintext/ciphertext can be publicly checked as $c = \mathcal{E}_{pk}(m)$. So, our method allows one to construct, for example, an efficient IND-CCA2 hybrid encryption scheme whose security relies on the hardness of inverting the RSA function or factoring large numbers.

In conclusion, our generic conversion may be seen as the best alternative to REACT when the underlying asymmetric encryption is relatively fast or when memory/bandwidth savings are a priority.

References

1. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. Full paper (30 pages), February 1999. An extended abstract appears in H. Krawczyk, ed., Advances in Cryptology – CRYPTO '98, volume 1462 of Lecture Notes in Computer Science, pages 26–45, Springer-Verlag, 1998.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
3. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.
4. Victor Boyko. On the security properties of OAEP as an all-or-nothing transform. Full paper (28 pages), August 1999. An extended abstract appears in M. Wiener, ed., Advances in Cryptology – CRYPTO '99, volume 1666 of Lecture Notes in Computer Science, pages 503–518, Springer-Verlag, 1999.
5. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. *IEICE Transaction on of Fundamentals of Electronic Communications and Computer Science* **E83-A**(1): 24–32, January 2000.
6. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.
7. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274. Springer-Verlag, 2001.
8. Oded Goldreich. On the foundations of modern cryptography. In B. Kaliski, editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 46–74. Springer-Verlag, 1997.
9. Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudo-randomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, 1999.
10. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
11. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM Annual Symposium on the Theory of Computing (STOC '90)*, pages 427–437. ACM Press, 1990.

12. Tatsuaki Okamoto and David Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175. Springer-Verlag, 2001.
13. David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 129–146. Springer-Verlag, 2000.
14. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.
15. Ronald L. Rivest. All-or-nothing encryption and the package transform. In E. Biham, editor, *Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 210–218. Springer-Verlag, 1997.

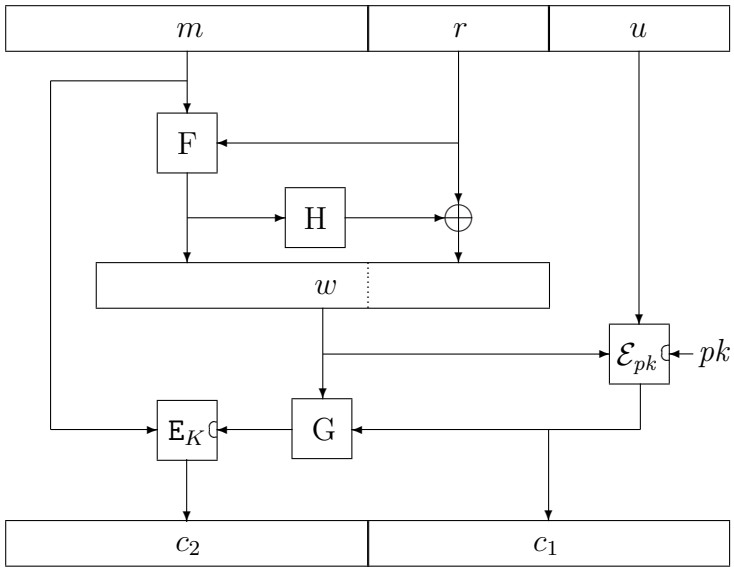


Fig. 1. Description of GEM in encryption mode.

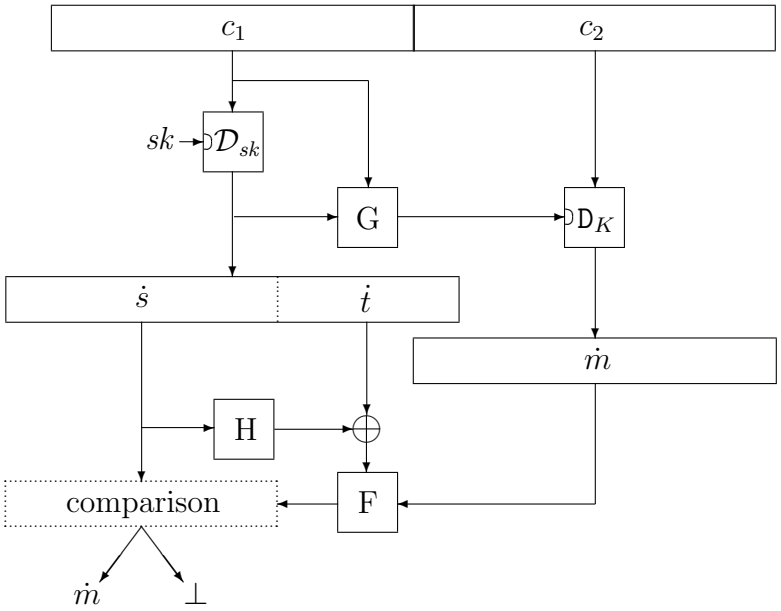


Fig. 2. Description of GEM in decryption mode.

Optimal Chosen-Ciphertext Secure Encryption of Arbitrary-Length Messages
(Extended Abstract)*

[Published in D. Naccache and Pascal Paillier, Eds., *Public Key Cryptography*, vol. 2274 of *Lecture Notes in Computer Science*, pp. 17–33, Springer-Verlag, 2002.]

Jean-Sébastien Coron[1], Helena Handschuh[1], Marc Joye[2], Pascal Paillier[1], David Pointcheval[3], and Christophe Tymen[1,3]

[1] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{jean-sebastien.coron, helena.handschuh, pascal.paillier, christophe.tymen}@gemplus.com
[2] Gemplus Card International
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos Cedex, France
marc.joye@gemplus.com − http://www.geocities.com/MarcJoye/
[3] École Normale Supérieure, Computer Science Department
45 rue d'Ulm, 75230 Paris Cedex 05, France
david.pointcheval@ens.fr − http://www.di.ens.fr/~pointche/

Abstract. This paper considers arbitrary-length chosen-ciphertext secure asymmetric encryption, thus addressing what is actually needed for a practical usage of strong public-key cryptography in the real world. We put forward two generic constructions, GEM-1 and GEM-2, which apply to explicit *fixed-length* weakly secure primitives and provide a strongly secure (IND-CCA2) public-key encryption scheme for messages of unfixed length (typically computer files). Our techniques optimally combine a single call to any one-way trapdoor function with repeated encryptions through some weak block-cipher (a simple XOR is fine) and hash functions of fixed-length input so that a minimal number of calls to these functions is needed. Our encryption/decryption throughputs are comparable to the ones of standard methods (asymmetric encryption of a session key + symmetric encryption with multiple modes). In our case, however, we *formally* prove that our designs are secure in the strongest sense and provide complete security reductions holding in the random oracle model.

1 Introduction

A real-life usage of public-key encryption requires three distinct ideal properties. *Security* is a major concern: a cryptosystem should be secure against any attack

* The full paper is available at <http://eprint.iacr.org/2002/011/>.

of any kind, should the attack be realistic in the context of use or only theoretical. *Performance* has to be risen to upmost levels to guarantee high speed encryption *and* decryption rates in communication protocols and real time applications. At last, *design simplicity* is desirable to save time and efforts in software or hardware developments, increase modularity and reusability, and facilitate public understanding and scrutiny.

Designing an encryption scheme which meets these criteria is quite a challenging work, but methodologies and tools exist, at least for the first property. Our knowledge of security features inherent to cryptographic objects and of the relations connecting them has intensively evolved lately, driving us to a growing range of powerful generic constructions, both simple and provably secure [FO99a, FO99b, Poi00, OP01b]. Among these constructions, Okamoto and Pointcheval's REACT [OP01b] is certainly the one that offers most flexibility: unlike Bellare and Rogaway's long-lived OAEP [BR95], REACT applies to any trapdoor function, *i.e.* any asymmetric encryption scheme presenting such a weak level of security as being OW-PCA (see further), to provide a cryptosystem of strongest level IND-CCA2 in the random oracle model.

1.1 Our results

This paper considers arbitrary-length chosen-ciphertext secure (IND-CCA2) asymmetric encryption schemes, thus addressing what is actually needed for a practical usage of strong public-key cryptography in the real world. We propose two generic constructions which apply to fixed-length weakly secure primitives and provide a strongly secure public-key cryptosystem for messages of unfixed length, such as computers files or communication streams. In our schemes, the encryption and decryption processes may start and progress without even knowing the overall input blocklength; they also may stop at any time. Besides, our designs are one-pass only, meaning that each message block will be treated exactly once.

Our techniques combine a *single* call to any one-way trapdoor function with repeated encryptions through some weak block-cipher (a simple XOR will do) and hash functions of fixed-length input. Contrarily to previous generic conversions, each message block will require only *one* call to a hash function so that the overall execution cost for an n-block plaintext is exactly 1 call to the one-way trapdoor encryption, followed by n calls to the block-cipher and $n + 1$ (or $n + 2$) calls to a hash function[1]. Besides, the storage of the whole plaintext file in memory is completely unnecessary and encryption/decryption procedures use a memory buffer of only ~ 3 blocks, thus allowing on-the-fly treatments of communication streams. We believe that our schemes are the first that combine these practical properties simultaneously while keeping total genericity.

The first construction applies to any OW-PCA probabilistic trapdoor function and incorporates two extra fields of fixed length in the ciphertext, one at each

[1] an extra hash call is needed for authenticity.

end. The second construction we give only works for deterministic OW trapdoor functions but adds only one extra field at the end of the ciphertext. Our performances are similar to the ones of standard methods, which usually encrypt some random session key under an asymmetric scheme and then feed that key into some block-cipher running under an appropriate multiple mode. In our case, however, we can formally prove that our designs are secure in the strongest sense IND-CCA2. Indeed, we provide complete security reductions holding in the random oracle model.

1.2 Outline of the paper

The paper is organized as follows. Section 2 briefly recalls security notions for encryption schemes in both symmetric and asymmetric settings. We also review Okamoto and Pointcheval's plaintext checking attacks in connection with computational gap problems [OP01a]. Then, Sections 3.1 and 3.2 introduce our new generic conversions, GEM-1 and GEM-2, whose reduction proofs are given in the extended version of this paper [CHJ+01]. Furthermore, typical examples of practical usage of these systems are given in Section 4. We conclude by giving some possible extensions of our work in Section 5.

2 Security notions for encryption schemes

2.1 Asymmetric encryption

We now introduce a few standard notations. An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Adversarial goals.

One-wayness. The first secrecy notion required from an encryption scheme is its *one-wayness*, meaning that one should not be able to recover a plaintext given its encryption. More formally, the scheme is said to be (τ, ε)-OW if for any adversary $\mathcal{A}$ with running time bounded by τ, the probability that $\mathcal{A}$ inverts $\mathcal{E}$ is less than ε:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr_{\substack{m \xleftarrow{R} \mathcal{M} \\ u \xleftarrow{R} \mathcal{U}}} [(sk, pk) \leftarrow \mathcal{K}(1^\kappa) : \mathcal{A}(\mathcal{E}_{pk}(m, u)) = m] < \varepsilon \ ,$$

where the probability is taken over the random choices of the adversary.

SEMANTIC SECURITY. Formalizing another security criterion that an encryption scheme should verify beyond one-wayness, Goldwasser and Micali [GM84] introduced the notion of *semantic security*. Also called *indistinguishability of encryptions* (or IND for short), this property captures the idea that an adversary should not be able to learn any information whatsoever about a plaintext, its length excepted, given its encryption. More formally, an asymmetric encryption scheme is said to be (τ, ε)-IND if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time upper-bounded by τ,

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{b \xleftarrow{R} \{0,1\} \\ u \xleftarrow{R} \mathcal{U}}} \left[\begin{array}{l} (sk, pk) \leftarrow \mathcal{K}(1^\kappa), (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathcal{E}_{pk}(m_b, u) : \mathcal{A}_2(c, \sigma) = b \end{array} \right] - 1 < \varepsilon \ ,$$

where the probability is taken over the random choices of $\mathcal{A}$. The two plaintexts m_0 and m_1 chosen by the adversary in $\mathcal{M}$ have to be of identical length.

NON-MALLEABILITY. The property of *non-malleability* (NM), independently proposed by Dolev, Dwork and Naor [DDN00], supposes that, given the encryption of a plaintext m, the attacker cannot produce the encryption of a related plaintext m'. Here, rather than learning some information about m, the adversary will try to output the encryption of m'. These two properties are related in the sense that non-malleability implies semantic security for any adversarial model, as pointed out in [DDN00] and [BDPR99].

Adversarial models. On the other hand, there exist several types of adversaries, or attack models. In a chosen-plaintext attack (CPA), the adversary has access to an encryption oracle, hence to the encryption of any plaintext she wants. Clearly, in the public-key setting, this scenario cannot be avoided. Naor and Yung [NY90] considered non-adaptive chosen-ciphertext attacks (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Finally, Rackoff and Simon [RS92] defined adaptive chosen-ciphertext attacks (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario.

Various security levels are then defined by pairing each goal (OW, IND or NM) with an attack model (CPA, CCA1 or CCA2), these two characteristics being considered separately. Interestingly, it has been shown that IND-CCA2 and NM-CCA2 were strictly equivalent notions [BDPR99]. This level is now considered as standard and referred to as IND-CCA2 security or chosen-ciphertext security. The security of a cryptosystem is thus measured as the ability to resist an adversarial goal in a given adversarial model. Whenever possible, the scheme is proven IND-CCA2 secure by exhibiting a polynomial reduction: if some adversary can break

the IND-CCA2 security of the system, then the same adversary can be invoked (polynomially many times) to solve some related hard problem.

2.2 Symmetric encryption schemes

A symmetric encryption scheme with key bit-length k and message bit-length m is a pair of algorithms $(\mathtt{E}, \mathtt{D})$ where

- $\mathtt{E}$ is a deterministic encryption algorithm which takes a key $k \in \{0,1\}^{\mathsf{k}}$ and a plaintext $m \in \{0,1\}^{\mathsf{m}}$ and returns a ciphertext $c \in \{0,1\}^{\mathsf{m}}$,
- $\mathtt{D}$ is a deterministic decryption algorithm which takes a key $k \in \{0,1\}^{\mathsf{k}}$ and a ciphertext $c \in \{0,1\}^{\mathsf{m}}$ and returns a plaintext $m \in \{0,1\}^{\mathsf{m}}$. We require that $\mathtt{D}_k(\mathtt{E}_k(m)) = m$ for all $m \in \{0,1\}^{\mathsf{m}}$ and $k \in \{0,1\}^{\mathsf{k}}$.

In this setting, again, various security notions are defined; most are adaptations from the asymmetric notions. In this work, however, we only need to define indistinguishability. A symmetric encryption scheme is said (τ, ε)-IND if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time bounded by τ,

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{k \xleftarrow{R} \{0,1\}^{\mathsf{k}} \\ b \xleftarrow{R} \{0,1\}}} [(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(\mathsf{k}), c \leftarrow \mathtt{E}_k(m_b) : \mathcal{A}_2(c, \sigma) = b] - 1 < \varepsilon \ ,$$

where the probability is also taken over the random choices of $\mathcal{A}$. Both plaintexts m_0 and m_1 are chosen by the adversary in $\{0,1\}^{\mathsf{k}}$. Although other attack scenarios may be considered, passive attacks are enough for our purposes. Note that this notion is a very weak requirement. Note also that the one-time pad encryption is perfectly indistinguishable, *i.e.* , it is $(\tau, 0)$-IND for any τ.

2.3 Plaintext-checking security

Okamoto and Pointcheval recently introduced an intermediate adversarial model called plaintext checking attacks [OP01b]. In this model, the adversary has access to a *plaintext-checking* oracle $\mathcal{O}^{\text{PCA}}$ which detects plaintext-ciphertext correspondences: the oracle takes as input a pair (m, c) and tells whether c encrypts m or not. Clearly, this oracle remains weaker than a decryption oracle because it is generally easier to check the solution of a problem (scheme inversion here) than to compute it. Obviously in the case of a deterministic encryption scheme, PCA and CPA are strictly equivalent attack scenarios. More specifically, any trapdoor permutation is OW-PCA if and only if it is OW (e.g., RSA).

From a complexity viewpoint, breaking a scheme's OW-PCA-security exactly consists in breaking its OW-security (*i.e.* its one-wayness) with the help of an oracle solving a weaker problem. That kind of problems, *i.e.* solving P_1 with access to $\mathcal{O}^{\mathrm{P}_2}$ and $\mathrm{P}_2 \Leftarrow \mathrm{P}_1$, are called *gap problems* [OP01a] and define some notion of complexity distance between problems in a hierarchy.

A typical example is ElGamal encryption, for which breaking OW is equivalent to CDH and having access to $\mathcal{O}^{\text{PCA}}$ allows to solve DDH trivially (and

conversely). OW-PCA-security is in this case equivalent to the gap problem separating CDH from DDH, which is called *Gap Diffie-Hellman Problem* and noted GDH (see [OP01a] for insights).

2.4 Generic conversions

In [BR95], Bellare and Rogaway proposed OAEP, a specific hash-based treatment applicable to any *partial-domain* [Sho01, FOPS01] one-way trapdoor permutation to provide an IND-CCA2 secure encryption scheme in the random oracle model [BR93]. Later, Fujisaki and Okamoto [FO99a] presented a way to transform, still in the random oracle model, any IND-PCA trapdoor function into an IND-CCA2 encryption scheme. They improved their results in [FO99b] where they gave a generic method to convert a *one-way* trapdoor function into an IND-CCA2 secure encryption scheme in the random oracle model[2]. A similar result was independently discovered by Pointcheval [Poi00]. More recently, Okamoto and Pointcheval [OP01b] proposed a more efficient generic conversion, called REACT. Contrarily to [FO99a, FO99b, Poi00], a complete re-encryption is unnecessary in the decryption process of REACT to ensure IND-CCA2 security, thus yielding a low running time overhead. Besides, REACT applies to any trapdoor function *i.e.* any asymmetric encryption scheme presenting such a weak level of security as being OW-PCA. Until now, however, no generic conversion has been explicitly defined[3] to encrypt messages of variable length based on fixed-length functions. The next section describes our arbitrary-length generic conversions.

3 Arbitrary-length IND-CCA2 encryption

The most popular and usual way of ensuring confidentiality of unfixed-length messages consists in public-key encrypting a random session key and then encrypting the message under that session key by the means of a block-cipher used within a suitable encryption mode. This approach has never been shown secure; in particular, the use of an IND-CCA2 asymmetric scheme to encrypt the session key is obviously insufficient to ensure any security whatsoever about the whole construction.

In comparison, our conversions are based on the same primitives, *i.e.* some asymmetric scheme $\mathcal{E}_{pk}$ and some symmetric scheme E_k. But we additionally use hash functions to make the session key evolve permanently as the encryption progresses. Our important result here is that the two cryptosystems we propose are IND-CCA2-secure provided that $\mathcal{E}_{pk}$ is OW-PCA or OW and E_k is indistinguishable. Independently, they provide different security/performance tradeoffs that we analyze in section 4.

[2] the conversion cost is however quite heavy as a complete re-encryption is needed during decryption.

[3] note that [OP01b] considers the case of variable-length encryption without providing any explicit construction for fixed-length functions.

3.1 Relying on a OW-PCA trapdoor function: GEM-1

Our first construction $\mathbb{E}^1_{pk}$ applies to any OW-PCA probabilistic trapdoor function $\mathcal{E}_{pk}$ and incorporates two extra fields of fixed length in the ciphertext, one at each end. To make the security proof easier, we will assume that the message blocklength is upper-bounded by some very large number n_{max} which value is discussed in section 4. The encryption and decryption procedures are as depicted below.

Encryption

Input plaintext $(m_1, \ldots, m_n)$, $1 \leq n \leq n_{max}$, random $\rho = w \| u$.
Output ciphertext $(t_1, c_1, \cdots, c_n, t_2)$ given by

$$\mathbb{E}^1_{pk}(m, \rho) = (\underbrace{\mathcal{E}_{pk}(w,u)}_{t_1}, \underbrace{\mathtt{E}_{k_1}(m_1)}_{c_1}, \underbrace{\mathtt{E}_{k_2}(m_2)}_{c_2}, \cdots, \underbrace{\mathtt{E}_{k_n}(m_n)}_{c_n}, \underbrace{\mathrm{F}(k_n, m_n, w)}_{t_2})$$

where $k_1 = \mathrm{H}_1(w, t_1)$, $k_2 = \mathrm{H}_2(k_1, m_1, w)$, $\ldots$, $k_n = \mathrm{H}_n(k_{n-1}, m_{n-1}, w)$.

Decryption

Input ciphertext $(t_1, c_1, \cdots, c_n, t_2)$ with $1 \leq n \leq n_{max}$.
Output plaintext $(\hat{m}_1, \cdots, \hat{m}_n)$ or $\perp$ according to

$$\mathbb{D}^1_{sk}(t_1, c_1, \cdots, c_n, t_2) = \begin{cases} \hat{m}_1 = \mathtt{D}_{\hat{k}_1}(c_1), \ldots, \hat{m}_n = \mathtt{D}_{\hat{k}_n}(c_n) & \text{if } t_2 = \mathrm{F}(\hat{k}_n, \hat{m}_n, \hat{w}) \\ \perp & \text{otherwise} \end{cases}$$

where $\hat{w} = \mathcal{D}_{sk}(t_1)$, $\hat{k}_1 = \mathrm{H}_1(\hat{w}, t_1)$ and $\hat{k}_i = \mathrm{H}_i(\hat{k}_{i-1}, \hat{m}_{i-1}, \hat{w})$ for $i = 2, n$.

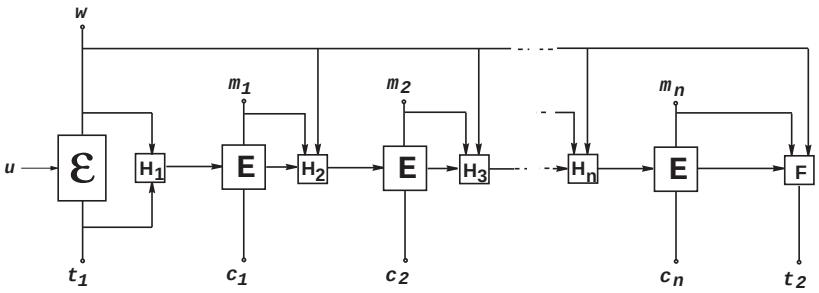


Fig. 1. Synopsis of GEM-1.

We claim that for any OW-PCA asymmetric encryption $\mathcal{E}_{pk}$ and any IND-secure symmetric encryption scheme $\mathtt{E}_k$, our converted scheme $\mathbb{E}^1_{pk}[\mathcal{E}_{pk}, \mathtt{E}_k]$ is IND-CCA2 in the random oracle model. To be more precise:

Theorem 1. *Suppose there exists an adversary $\mathcal{A}$ which distinguishes $\mathbb{E}^1_{pk}[\mathcal{E}_{pk}, \mathtt{E}_k]$ within a time bound τ with advantage ε in less than q_{F}, $q_{\mathrm{H}} = \sum_{i \in \langle 1, n_{max} \rangle} q_{\mathrm{H}_i}$,*

$q_{\mathbb{D}^1_{sk}}$ oracle calls. Suppose also that $\mathtt{E}_k$ is (τ,ν)-indistinguishable. Then there exists an algorithm $\mathcal{B}$ which inverts $\mathcal{E}_{pk}$ with probability ε' greater than

$$\varepsilon' \geq \frac{\varepsilon}{2} - q_{\mathbb{D}^1_{sk}} \left(\frac{1}{\sharp\, t_2} + \frac{3}{\sharp\, k} \right) - n_{max} \left(\frac{\nu}{2} + \frac{q_{\mathbb{D}^1_{sk}}}{\sharp\, k} \right) ,$$

with a total number of calls to $\mathcal{O}^{\text{PCA}}$ upper-bounded by $q_{\mathcal{O}^{\text{PCA}}} \leq q_{\text{F}} + q_{\text{H}}$ and in time

$$\tau_{\mathcal{B}} = \tau + (q_{\mathbb{D}^1_{sk}} + 1)\,(q_{\text{F}} + q_{\text{H}}) \cdot (\tau_{\text{PCA}} + O(1)) \ .$$

Here, $\sharp\, a$ denotes the number of all possible values of a (hence $\sharp\, k = 2^{\mathsf{k}}$).

We refer the reader to the (extensive) reduction proof given in the extended version of this work [CHJ+01].

3.2 Relying on a OW trapdoor function: GEM-2

Our second construction $\mathbb{E}^2_{pk}$ only works with a deterministic OW trapdoor function $\mathcal{E}_{pk}$ (such as RSA) but adds only one extra field at the end of the ciphertext. Here again, we will assume that the message blocklength is upper-bounded by some large number n_{max}. The encryption and decryption procedures follow.

Encryption

Input plaintext $(m_1, \ldots, m_n)$, $1 \leq n \leq n_{max}$, random r.
Output ciphertext $(c_1, \cdots, c_n, t)$ given by

$$\mathbb{E}^2_{pk}(m,r) = (\underbrace{\mathtt{E}_{k_1}(m_1)}_{c_1}, \underbrace{\mathtt{E}_{k_2}(m_2)}_{c_2}, \cdots, \underbrace{\mathtt{E}_{k_n}(m_n)}_{c_n}, \underbrace{\mathcal{E}_{pk}(s\|v)}_{t})$$

$$\text{where } \begin{cases} k_1 = \mathrm{G}_1(r), k_i = \mathrm{G}_i(k_{i-1}, m_{i-1}, r) \text{ for } i = 2, \ldots n, \\ s = \mathrm{F}(k_n, m_n, r), \text{ and } v = r \oplus \mathrm{H}(s). \end{cases}$$

Decryption

Input ciphertext $(c_1, \cdots, c_n, t)$ with $1 \leq n \leq n_{max}$.
Output plaintext $(\hat{m_1}, \cdots, \hat{m_n})$ or $\perp$ according to

$$\mathbb{D}^2_{sk}(c_1, \cdots, c_n, t) = \begin{cases} \hat{m}_1 = \mathtt{D}_{\hat{k}_1}(c_1), \ldots, \hat{m}_n = \mathtt{D}_{\hat{k}_n}(c_n) & \text{if } \hat{s} = \mathrm{F}(\hat{k}_n, \hat{m}_n, \hat{r}), \\ \perp & \text{otherwise.} \end{cases}$$

$$\text{where } \begin{cases} \hat{s}\|\hat{v} = \mathcal{D}_{sk}(t), \hat{r} = \hat{v} \oplus \mathrm{H}(\hat{s}), \\ \hat{k}_1 = \mathrm{G}_1(\hat{r}), \text{and } \hat{k}_i = \mathrm{G}_i(\hat{k}_{i-1}, \hat{m}_{i-1}, \hat{r}) \text{ for } i = 2, \ldots n. \end{cases}$$

We claim that for any OW asymmetric encryption $\mathcal{E}_{pk}$ and any IND-secure symmetric encryption scheme $\mathtt{E}_k$, the converted scheme $\mathbb{E}^2_{pk}[\mathcal{E}_{pk}, \mathtt{E}_k]$ is IND-CCA2 in the random oracle model. To be more precise:

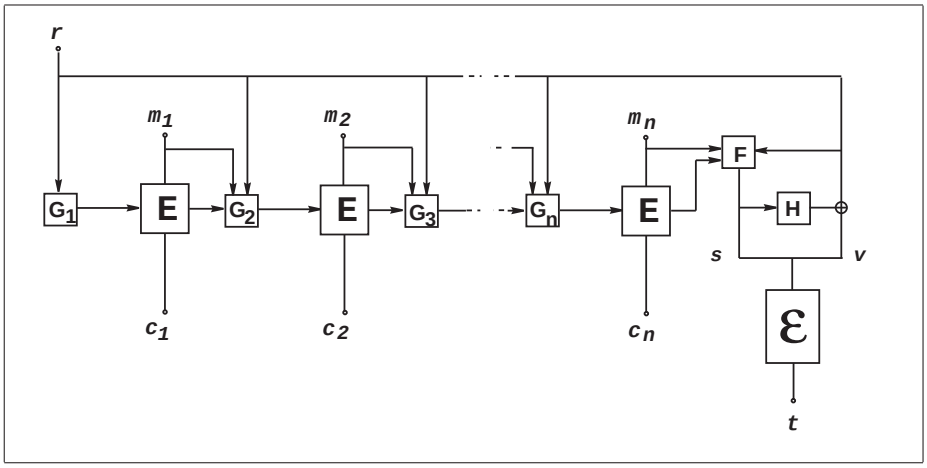


Fig. 2. Synopsis of GEM-2.

Theorem 2. *Suppose there exists an adversary $\mathcal{A}$ which distinguishes $\mathbb{E}^2_{pk}[\mathcal{E}_{pk}, \mathtt{E}_k]$ within a time bound τ with advantage ε in less than q_{F}, q_{H}, $q_{\mathrm{G}} = \sum_{i\in\langle 1,n_{max}\rangle} q_{\mathrm{G}_i}$, $q_{\mathbb{D}^2_{sk}}$ oracle calls. Suppose also that $\mathtt{E}_k$ is (τ,ν)-indistinguishable. Then there exists an algorithm $\mathcal{B}$ which inverts $\mathcal{E}_{pk}$ with probability ε' greater than*

$$\varepsilon' \geq \frac{\varepsilon}{2} - \frac{q_{\mathrm{F}}+q_{\mathrm{G}}}{\sharp\, r} - q_{\mathbb{D}^2_{sk}}(q_{\mathrm{F}}+1)\left(\frac{1}{\sharp\, s}+\frac{1}{\sharp\, r}\right) - \frac{q_{\mathbb{D}^2_{sk}}}{\sharp\, k} - n_{max}\left(\frac{\nu}{2}+\frac{q_{\mathbb{D}^2_{sk}}}{\sharp\, k}\right) ,$$

within a time bounded by

$$\tau_{\mathcal{B}} = \tau + (q_{\mathbb{D}^2_{sk}}+1)\,(q_{\mathrm{F}}+q_{\mathrm{G}})\,q_{\mathrm{H}}\cdot(\tau_{\mathcal{E}}+O(1)) ,$$

where $\tau_{\mathcal{E}}$ denotes the maximum time needed by $\mathcal{E}_{pk}$ for a single encryption.

Again, the reader is invited to find the reduction proof in [CHJ+01] for technical details.

4 Applications

Numerous applications are possible when embodying $\mathcal{E}_{pk}$ and $\mathtt{E}_k$. Due to lack of space, we will only consider the typical case $\mathcal{E}_{pk} = \mathrm{RSA}$ and $\mathtt{E}_k = \oplus$ (for which $\nu = 0$). The instantiations of random oracles F, H_i, H and G_i in one scheme or another by hash functions can be done by setting for instance $\mathrm{H}_i(\cdot) = \mathrm{SHA}(\cdot\|i)$ where the counter $i \in \langle 1, n_{max}\rangle$ is incremented at each block treatment. Special values of i such as 0 or -1 may be used to implement F and H.

4.1 $\mathbb{E}^{1}_{pk}[\mathbf{RSA}, \oplus]$

Corollary 1. *The encryption scheme* $\mathbb{E}^{1}_{pk}[\mathrm{RSA}, \oplus]$ *is* IND-CCA2 *in the random oracle model under the RSA assumption.*

For concrete security parameters, we suggest to use 1024-bit RSA keys with public exponent $e = \mathbb{F}_4 = 2^{16} + 1$. We set for instance $\log_2 \sharp\, t_2 = \mathsf{m} = \mathsf{k} = 160$ (hash functions F, H_i being derived from SHA-1 using a counter $i \in \langle 1, n_{max} \rangle$ like described above), $\sharp\, w = 2^{160}$ and $n_{max} = 2^{32}$. Assuming that the probability ε' to invert RSA lies around $\varepsilon' = 2^{-60}$, then an attaquer could distinguish $\mathbb{E}^{1}_{pk}[\mathrm{RSA}, \oplus]$ with $q_{\mathbb{D}^{1}_{sk}} = 2^{50}$ decryptions with advantage no more than $\varepsilon = 2^{-58}$.

From an implementation viewpoint, note that as soon as the RSA encryption has been done, the encryption procedure may directly output ciphertexts blocks one after the other without having to wait that all blocks are encrypted to transmit them all together. This allows on-the-fly encryption of communication streams. Three-tuples (w, y, k_1) may also be computed in advance to let the encryption device or software deal with hash computations only. The suggested setting allows to replace oracles $\mathrm{H}_2, \ldots, \mathrm{H}_n, \mathrm{F}$ by the compression function $(512 \mapsto 160)$ of SHA-1, driving us to $n + 3$ calls to this function since the input of H_1 is made of three 512-bit blocks. Another benefit of our construction is that it requires only a small memory buffer (one field for the storage of w, one for the current key k_i and a third one for m_i). Finally, hardware implementations providing some hash coprocessor may drastically increase our speed rates.

4.2 $\mathbb{E}^{2}_{pk}[\mathbf{RSA}, \oplus]$

Corollary 2. *The encryption scheme* $\mathbb{E}^{2}_{pk}[\mathrm{RSA}, \oplus]$ *is* IND-CCA2 *in the random oracle model under the RSA assumption.*

For concrete security bounds, the same suggestions as previously lead to a maximal advantage of $\varepsilon = 2^{-58}$ if we take $\log_2 \sharp\, s = \log_2 \sharp\, r = 512$, $q_{\mathrm{F}} = q_{\mathrm{G}} = 2^{50}$ and $n_{max} = 2^{32}$.

Here again, any smart implementation allows on-the-fly encryption. The memory requirements are similar to the one of $\mathbb{E}^{1}_{pk}$. Here too, a coprocessor devoted to hash computations would increase speed rates.

5 Conclusion

We devised new generic constructions which apply to fixed-length weakly secure primitives and provide a strongly secure (IND-CCA2) public-key encryption scheme for messages of unfixed length like computer files or communications streams. An open question resides in investigating whether simpler and/or faster designs could exist, or whether the security requirements on the primitives could be shrunk further. Another challenging topic would be to come up with a construction holding only one additional field in the ciphertext but still employing a probabilistic encryption $\mathcal{E}_{pk}$ as in $\mathbb{E}^{1}_{pk}$. Finally, one could try to include a

signature scheme in the encryption process to simultaneously authenticate the sender's identity, the plaintext and the ciphertext itself. Such an extension would ideally lead to fast and secure (according to one-more decryption attacks) signcryption schemes for arbitrary-length messages.

References

[BDPR99] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. Full paper (30 pages), February 1999. An extended abstract appears in H. Krawczyk, ed., Advances in Cryptology – CRYPTO '98, volume 1462 of Lecture Notes in Computer Science, pages 26–45, Springer-Verlag, 1998.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.

[CHJ+01] Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, and Christophe Tymen. Optimal chosen-ciphertext secure encryption of arbitrary-length messages. `www.gemplus.com`, 2001.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[FO99a] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer-Verlag, 1999.

[FO99b] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.

[FOPS01] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is secure under the rsa assumption. In *Advances in Cryptology – CRYPTO '01*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM Annual Symposium on the Theory of Computing (STOC '90)*, pages 427–437. ACM Press, 1990.

[OP01a] Tatsuaki Okamoto and David Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *PKC*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer-Verlag, 2001.

[OP01b] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In D. Naccache, editor, *RSA 2001 Cryptographers' Track*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175. Springer-Verlag, 2001.

[Poi00] David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 129–146. Springer-Verlag, 2000.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576, pages 433–444. Springer-Verlag, 1992.

[Sho01] Victor Shoup. OAEP reconsidered. In *Advances in Cryptology – CRYPTO '01*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

Universal Padding Schemes for RSA

[Published in M. Yung, Ed., *Advances in Cryptology – CRYPTO 2002*, vol. 2442 of *Lecture Notes in Computer Science*, pp. 226–241, Springer-Verlag, 2002.]

Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier

Gemplus Card International, France
{jean-sebastien.coron, marc.joye, david.naccache, pascal.paillier}@gemplus.com

Abstract. A common practice to encrypt with RSA is to first apply a padding scheme to the message and then to exponentiate the result with the public exponent; an example of this is OAEP. Similarly, the usual way of signing with RSA is to apply some padding scheme and then to exponentiate the result with the private exponent, as for example in PSS. Usually, the RSA modulus used for encrypting is different from the one used for signing. The goal of this paper is to simplify this common setting. First, we show that PSS can also be used for encryption, and gives an encryption scheme semantically secure against adaptive chosen-ciphertext attacks, in the random oracle model. As a result, PSS can be used indifferently for encryption or signature. Moreover, we show that PSS allows to safely use the same RSA key-pairs for both encryption and signature, in a concurrent manner. More generally, we show that using PSS the same set of keys can be used for both encryption and signature for any trapdoor partial-domain one-way permutation. The practical consequences of our result are important: PKIs and public-key implementations can be significantly simplified.

Keywords. Probabilistic Signature Scheme, Provable Security.

1 Introduction

A very common practice for encrypting a message m with RSA is to first apply a padding scheme μ, then raise $\mu(m)$ to the public exponent e. The ciphertext c is then

$$c = \mu(m)^e \bmod N \ .$$

Similarly, for signing a message m, the common practice consists again in first applying a padding scheme μ' then raising $\mu'(m)$ to the private exponent d. The signature s is then

$$s = \mu'(m)^d \bmod N \ .$$

For various reasons, it would be desirable to use the same padding scheme $\mu(m)$ for encryption and for signature: in this case, only one padding scheme needs to be implemented. Of course, the resulting padding scheme $\mu(m)$ should be provably secure for encryption and for signing. We say that a padding scheme is *universal* if it satisfies this property.

The strongest public-key encryption security notion was defined in [15] as *indistinguishability under an adaptive chosen ciphertext attack*. An adversary should not be able to distinguish between the encryption of two plaintexts, even if he can obtain the decryption of ciphertexts of his choice. For digital signature schemes, the strongest security notion was defined by Goldwasser, Micali and Rivest in [10], as *existential unforgeability under an adaptive chosen message attack*. This notion captures the property that an adversary cannot produce a valid signature, even after obtaining the signature of (polynomially many) messages of his choice.

In this paper, we show that the padding scheme PSS [3], which is originally a provably secure padding scheme for producing signatures, can also be used as a provably secure encryption scheme. More precisely, we show that PSS offers indistinguishability under an adaptive chosen ciphertext attack, in the random oracle model, under the partial-domain one-wayness of the underlying permutation. Partial-domain one-wayness, introduced in [9], is a formally stronger assumption than one-wayness. However, for RSA, partial-domain one-wayness is equivalent to (full domain) one-wayness and therefore RSA-PSS encryption is provably secure under the sole assumption that RSA is one-way.

Generally, in a given application, the RSA modulus used for encrypting is different from the RSA modulus used for signing; our setting (and real-world PKIs) would be further simplified if one could use the same set of keys for both encryption and signature (see [11]). In this paper, we show that using PSS, the same keys can be safely used for encryption and for signature.

2 Public-Key Encryption

A public-key encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of public and secret keys (pk, sk) depending on some security parameter k;
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes as input a public key pk and a plaintext $M \in \mathcal{M}$, runs on a random tape $r \in \mathcal{R}$ and returns a ciphertext c. $\mathcal{M}$ and $\mathcal{R}$ stand for spaces in which messages and random strings are chosen respectively;
- $\mathcal{D}$ is a deterministic decryption algorithm which, given as input a secret key sk and a ciphertext c, returns the corresponding plaintext M, or Reject.

The strongest security notion for public-key encryption is the aforementioned notion of indistinguishability under an adaptive chosen ciphertext attack. An adversary should not be able to distinguish between the encryption of two plaintexts, even if he can obtain the decryption of ciphertexts of his choice. The attack scenario is the following.

1. The adversary $\mathcal{A}$ receives the public key pk with $(pk, sk) \leftarrow \mathcal{K}(1^\kappa)$.
2. $\mathcal{A}$ makes decryption queries for ciphertexts y of his choice.
3. $\mathcal{A}$ chooses two messages M_0 and M_1 of identical length, and receives the encryption c of M_b for a random unknown bit b.
4. $\mathcal{A}$ continues to make decryption queries. The only restriction is that the adversary cannot request the decryption of c.
5. $\mathcal{A}$ outputs a bit b', representing its "guess" on b.

The adversary's advantage is then defined as

$$\mathsf{Adv}(\mathcal{A}) = |2 \cdot \Pr[b' = b] - 1| \ .$$

An encryption scheme is said to be secure against adaptive chosen ciphertext attack (and denoted IND-CCA2) if the advantage of any polynomial-time bounded adversary is a negligible function of the security parameter. Usually, schemes are proven to be IND-CCA2 secure by exhibiting a polynomial reduction: if some adversary can break the IND-CCA2 security of the system, then the same adversary can be invoked (polynomially many times) to solve a related hard problem.

The random oracle model, introduced by Bellare and Rogaway in [1], is a theoretical framework in which any hash function is seen as an oracle which outputs a random value for each new query. Actually, a security proof in the random oracle model does not necessarily imply that a scheme is secure in the real world (see [6]). Nevertheless, it seems to be a good engineering principle to design a scheme so that it is provably secure in the random oracle model. Many encryption and signature schemes were proven to be secure in the random oracle model.

3 Encrypting with PSS-R

In this section we prove that given any trapdoor partial-domain one-way permutation f, the encryption scheme defined by first applying PSS with message recovery (denoted PSS-R) and then encrypting the result with f achieves the strongest security level for an encryption scheme, in the random oracle model.

3.1 The PSS-R padding scheme

PSS-R, defined in [3], is parameterized by the integers k, k_0 and k_1 and uses two hash functions

$$H : \{0,1\}^{k-k_1} \to \{0,1\}^{k_1} \quad \text{and} \quad G : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1} \ .$$

PSS-R takes as input a $(k-k_0-k_1)$-bit message M and a k_0-bit random integer r. As illustrated in Figure 1, PSS-R outputs

$$\mu(M, r) = \omega || s$$

where $\|$ stands for concatenation, $\omega = H(M||r)$ and $s = G(\omega) \oplus (M \| r)$. Actually, in [3], $M \| r$ is used as the argument to H and $r \| M$ is used as the mask to xor with $G(\omega)$. Here for simplicity we use $M \| r$ in both places, but the same results apply either way.

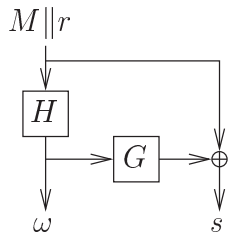


Fig. 1. The PSS-R padding scheme.

3.2 The PSS-E encryption scheme

The new encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, that we denote PSS-E, is based on μ and a k-bit trapdoor permutation f:

- $\mathcal{K}$ generates the public key f and the secret key f^{-1};
- $\mathcal{E}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random $r \in \{0,1\}^{k_0}$, the encryption algorithm outputs the ciphertext

$$c = f(\mu(M, r))$$

- $\mathcal{D}(c)$: the decryption algorithm recovers $(\omega, s) = f^{-1}(c)$ and then $M\|r = G(\omega) \oplus s$. If $\omega = H(M||r)$, the algorithm returns M, otherwise it returns Reject.

3.3 The underlying problem

The security of PSS-E is based on the difficulty of inverting f without knowing f^{-1}. As in [9], we use two additional related problems: the partial-domain one-wayness and the set partial-domain one-wayness of f:

- **(τ, ε)-one-wayness of f**, means that for any adversary $\mathcal{A}$ who wishes to recover the full pre-image (ω, s) of $f(\omega, s)$ in time less than τ, $\mathcal{A}$'s success probability $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$ is upper-bounded by ε:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr_{\omega,s}[\mathcal{A}(f(\omega, s)) = (\omega, s)] < \varepsilon$$

- **(τ, ε)-partial-domain one-wayness of f**, means that for any adversary $\mathcal{A}$ who wishes to recover the partial pre-image ω of $f(\omega, s)$ in time less than τ, $\mathcal{A}$'s success probability $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$ is upper-bounded by ε:

$$\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A}) = \Pr_{\omega,s}[\mathcal{A}(f(\omega, s)) = \omega] < \varepsilon$$

- **$(\ell, \tau, \varepsilon)$-set partial-domain one-wayness of f**, means that for any adversary $\mathcal{A}$ who wishes to output a set of ℓ elements which contains the

partial pre-image ω of $f(\omega, s)$, in time less than τ, $\mathcal{A}$'s success probability $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A})$ is upper-bounded by ε:

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A}) = \Pr_{\omega,s}[\omega \in \mathcal{A}(f(\omega, s))] < \varepsilon \ .$$

As in [9], we denote by $\mathsf{Succ}^{\mathsf{ow}}(\tau)$, (resp. $\mathsf{Succ}^{\mathsf{pd-ow}}(\tau)$ and $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau)$) the maximal probability $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$, (resp. $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$ and $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A})$), over all adversaries whose running times are less than τ. For any τ and $\ell \geq 1$, we have

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau) \geq \mathsf{Succ}^{\mathsf{pd-ow}}(\tau) \geq \mathsf{Succ}^{\mathsf{ow}}(\tau) \ .$$

Moreover, by randomly selecting any element in the set returned by the adversary against the set partial-domain one-wayness, one can break the partial-domain one-wayness with probability $1/\ell$, which gives

$$\mathsf{Succ}^{\mathsf{pd-ow}}(\tau) \geq \mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau)/\ell \ . \tag{1}$$

We will see in Section 5 that for RSA, the three problems are polynomially equivalent.

3.4 Security of PSS-E

The following theorem shows that PSS-E is semantically secure under adaptive chosen ciphertext attacks (IND-CCA2), in the random oracle model, assuming that the underlying permutation is partial-domain one-way.

Theorem 1. *Let $\mathcal{A}$ be a CCA2-adversary against the semantic security of PSS-E, with advantage ε and running time t, making q_D, q_H and q_G queries to the decryption oracle and the hash functions H and G, respectively. Then*

$$\mathsf{Succ}^{\mathsf{pd-ow}}(t') \geq \frac{1}{q_H + q_G} \cdot \left(\varepsilon - q_H 2^{-k_0} - q_D 2^{-k_1}\right)$$

where $t' \leq t + q_H \cdot T_f$, and T_f denotes the time complexity of f.

The theorem follows from inequality (1) and the next lemma.

Lemma 1. *Using the notations introduced in Theorem 1, we have:*

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H + q_G, t') \geq \varepsilon - q_H \cdot 2^{-k_0} - q_D \cdot 2^{-k_1} \tag{2}$$

Proof. We describe a reduction $\mathcal{B}$ which using $\mathcal{A}$, constructs an adversary against the set partial-domain one-wayness of f. We start with a top-level description of the reduction and then show how to simulate the random oracles G, H and the decryption oracle D. Eventually we compute the success probability of $\mathcal{B}$.

Top-level description of the reduction $\mathcal{B}$.

1. $\mathcal{B}$ is given a function f and $c^* = f(\omega^*, s^*)$, for random integers ω^* and s^*. $\mathcal{B}$'s goal is to output a list which contains the partial pre-image ω^* of c^*.
2. $\mathcal{B}$ runs $\mathcal{A}$ with f and gets $\{M_0, M_1\}$. It chooses a random bit b and gives c^* as a ciphertext for M_b. $\mathcal{B}$ simulates the oracles G, H and D as described below.
3. $\mathcal{B}$ receives from $\mathcal{A}$ the answer b' and outputs the list of queries asked to G.

Simulation of the random oracles G, H and D. The simulation of G and H is very simple: a random answer is returned for each new query of G and H. Moreover, when ω is the answer of a query to H, we simulate a query for ω to G, so that $G(\omega)$ is defined.

On query c to the decryption oracle, the reduction $\mathcal{B}$ looks at each query $M'||r'$ to H and computes

$$\omega' = H(M'||r') \text{ and } s' = G(\omega') \oplus (M'\|r')$$

Then if $c = f(\omega', s')$ the reduction $\mathcal{B}$ returns M'. Otherwise, the reduction outputs Reject.

Analysis. Since $c^* = f(\omega^*, s^*)$ is the ciphertext corresponding to M_b, we have the following constraint for random oracles G and H:

$$H(M_b\|r^*) = \omega^* \quad \text{and} \quad G(\omega^*) = s^* \oplus (M_b\|r^*) \ . \tag{3}$$

We denote by AskG the event: "ω^* has been asked to G" and by AskH the event: "there exists M' such that $M'||r^*$ has been queried to H".

If ω^* was never queried to G, then $G(\omega^*)$ is undefined and r^* is then a uniformly distributed random variable. Therefore the probability that there exists M' such that (M', r^*) has been asked to H is at most $q_H \cdot 2^{-k_0}$. This gives

$$\Pr[\mathsf{AskH}|\neg\mathsf{AskG}] \leq q_H \cdot 2^{-k_0} \ . \tag{4}$$

Our simulation of D can only fail by rejecting a valid ciphertext. We denote by DBad this event. Letting $c = f(\omega, s)$ be the ciphertext queried to D and

$$M\|r = G(\omega) \oplus s$$

we reject a valid ciphertext if $H(M||r) = \omega$ while $M||r$ was never queried to H. However, if $M\|r$ was never queried to H, then $H(M\|r)$ is randomly defined. Namely if the decryption query occured before c^* was sent to the adversary, then constraint (3) does not apply and $H(M\|r)$ is randomly defined. Otherwise, if the decryption query occured after c^* was sent to the adversary, then $c \neq c^*$ implies $(M, r) \neq (M_b, r^*)$ and $H(M\|r)$ is still randomly defined. In both cases the probability that $H(M, r) = \omega$ is then 2^{-k_1}, which gives

$$\Pr[\mathsf{DBad}] \leq q_D \cdot 2^{-k_1} \ . \tag{5}$$

Let us denote by Bad the event: "ω^* has been queried to G or (M', r^*) has been queried to H for some M' or the simulation of D has failed". Formally

$$\mathsf{Bad} = \mathsf{AskG} \vee \mathsf{AskH} \vee \mathsf{DBad} \tag{6}$$

Let us denote by S the event: "the adversary outputs the correct value for b, i.e., $b = b'$". Conditioned on $\neg\mathsf{Bad}$, our simulations of G, H and D are independent of b, and therefore $\mathcal{A}$'s view is independent of b as well. This gives:

$$\Pr[\mathsf{S}|\neg\mathsf{Bad}] = \frac{1}{2} \ . \tag{7}$$

Moreover, conditioned on $\neg\mathsf{Bad}$, the adversary's view is the same as when interacting with (perfect) random and decryption oracles, which gives

$$\Pr[\mathsf{S} \wedge \neg\mathsf{Bad}] \geq \frac{1}{2} + \frac{\varepsilon}{2} - \Pr[\mathsf{Bad}] \ . \tag{8}$$

From (7) we obtain

$$\Pr[\mathsf{S} \wedge \neg\mathsf{Bad}] = \Pr[\mathsf{S}|\neg\mathsf{Bad}] \cdot \Pr[\neg\mathsf{Bad}] = \frac{1}{2}(1 - \Pr[\mathsf{Bad}])$$

which gives using (8)

$$\Pr[\mathsf{Bad}] \geq \varepsilon \ . \tag{9}$$

From (6) we have

$$\begin{aligned} \Pr[\mathsf{Bad}] &\leq \Pr[\mathsf{AskG} \vee \mathsf{AskH}] + \Pr[\mathsf{DBad}] \\ &\leq \Pr[\mathsf{AskG}] + \Pr[\mathsf{AskH} \wedge \neg\mathsf{AskG}] + \Pr[\mathsf{DBad}] \\ &\leq \Pr[\mathsf{AskG}] + \Pr[\mathsf{AskH}|\neg\mathsf{AskG}] + \Pr[\mathsf{DBad}] \end{aligned}$$

which yields using (4), (5) and (9):

$$\Pr[\mathsf{AskG}] \geq \varepsilon - q_H \cdot 2^{-k_0} - q_D \cdot 2^{-k_1}$$

and hence (2) holds. This completes the proof of Lemma 1. □

4 Signing and Encrypting with the Same Set of Keys

In this section we show that when using PSS, the *same* public key can be used for encryption and signature in a concurrent manner. For RSA, this means that the same set (N, e, d) can be used for both operations. In other words, when Alice sends a message to Bob, she encrypts it using Bob's public key (N, e); Bob decrypts it using the corresponding private key (N, d). To sign a message M, Bob will use the *same* private key (N, d). As usual, anybody can verify Bob's signatures using his public pair (N, e).

Although provably secure (as we will see hereafter), this is contrary to the folklore recommendation that signature and encryption keys should be distinct.

This recommendation may prove useful is some cases; this is particularly true when a flaw has been found in the encryption scheme or in the signature scheme. In our case, we will prove that when using PSS-R, a decryption oracle does not help the attacker in forging signatures, and a signing oracle does not help the attacker in gaining information about the plaintext corresponding to a ciphertext.

Nevertheless, we advise to be very careful when implementing systems using the same keys for encrypting and signing. For example, if there are some implementation errors in a decryption server (see for example [13]), then an attacker could use this server to create forgeries.

4.1 The PSS-ES encryption and signature scheme

The PSS-ES encryption and signature scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$ is based on PSS-R and a k-bit trapdoor permutation f. As for the PSS-R signature scheme, the signature scheme in PSS-ES is with message recovery: this means that the message is recovered when verifying the signature. In this case, only messages of fixed length $k - k_0 - k_1$ can be signed. To sign messages M of arbitrary length, it suffices to apply a collision-free hash function to M prior to signing.

- $\mathcal{K}$ generates the public key f and the secret key f^{-1};
- $\mathcal{E}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random value $r \in \{0,1\}^{k_0}$, the encryption algorithm computes the ciphertext

$$c = f(\mu(M, r))$$

- $\mathcal{D}(c)$: the encryption algorithm recovers $(\omega, s) = f^{-1}(c)$ and computes

$$M \| r = G(\omega) \oplus s \ .$$

  If $\omega = H(M||r)$, the algorithm returns M, otherwise it returns Reject;
- $\mathcal{S}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random value $r \in \{0,1\}^{k_0}$, the signing algorithm computes the signature

$$\sigma = f^{-1}(\mu(M, r))$$

- $\mathcal{V}(\sigma)$: given the signature σ, the verification algorithm recovers $(\omega, s) = f(\sigma)$ and computes:

$$M \| r = G(\omega) \oplus s \ .$$

  If $\omega = H(M\|r)$, the algorithm accepts the signature and returns M. Otherwise, the algorithm returns Reject.

4.2 Semantic security

We must ensure that an adversary is still unable to distinguish between the encryption of two messages, even if he can obtain the decryption of ciphertexts of his choice, and the signature of messages of his choice. The attack scenario is

consequently the same as previously, except that the adversary can also obtain the signature of messages he wants.

The following theorem, whose proof is given in Appendix A, shows that PSS-ES is semantically secure under adaptive chosen ciphertext attacks, in the random oracle model, assuming that the underlying permutation is partial domain one-way.

Theorem 2. *Let $\mathcal{A}$ be an adversary against the semantic security of PSS-ES, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash functions H and G, respectively. Then, $\mathsf{Succ}^{\mathsf{pd-ow}}(t')$ is greater than*

$$\frac{1}{q_H + q_G + q_{sig}} \left(\varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - q_D 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1}\right)$$

where $t' \leq t + (q_H + q_{sig}) \cdot T_f$, and T_f denotes the time complexity of f.

4.3 Unforgeability

For signature schemes, the strongest security notion is the previously introduced existential unforgeability under an adaptive chosen message attack. An attacker cannot produce a valid signature, even after obtaining the signature of (polynomially many) messages of his choice. Here the adversary can also also obtain the decryption of ciphertexts of his choice under the same public-key. Consequently, the attack scenario is the following.

1. The adversary $\mathcal{A}$ receives the public key pk with $(pk, sk) \leftarrow \mathcal{K}(1^\kappa)$.
2. $\mathcal{A}$ makes signature queries for messages M of his choice. Additionally, he makes decryption queries for ciphertexts y of his choice.
3. $\mathcal{A}$ outputs the signature of a message M' which was not queried for signature before.

An encryption-signature scheme is said to be secure against chosen-message attacks if for any polynomial-time bounded adversary, the probability to output a forgery is negligible.

The following theorem shows that PSS-ES is secure against an adaptive chosen message attack. The proof is similar to the security proof of PSS [3] and is given in Appendix B.

Theorem 3. *Let $\mathcal{A}$ be an adversary against the unforgeability of PSS-ES, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash oracles H and G, respectively. Then $\mathsf{Succ}^{\mathsf{ow}}(t')$ is greater than*

$$\frac{1}{q_H} \left(\varepsilon - ((q_H + q_{sig})^2 + q_D + 1) \cdot 2^{-k_1}\right) \tag{10}$$

where $t' \leq t + (q_H + q_{sig}) \cdot T_f$, and T_f denotes the time complexity of f.

5 Application to RSA

5.1 The RSA cryptosystem

The RSA cryptosystem [16] is the most widely used cryptosystem today. In this section, we show that by virtue of RSA's homomorphic properties, the partial-domain one-wayness of RSA is equivalent to the one-wayness of RSA. This enables to prove that the encryption scheme RSA-PSS-E and the encryption and signature scheme RSA-PSS-ES are semantically secure against chosen ciphertext attacks, in the random oracle model, assuming that inverting RSA is hard.

Definition 1 (The RSA Primitive). *The RSA primitive is a family of trapdoor permutations, specified by:*

- *The RSA generator $\mathcal{RSA}$, which on input 1^k, randomly selects two distinct $k/2$-bit primes p and q and computes the modulus $N = p \cdot q$. It randomly picks an encryption exponent $e \in \mathbb{Z}^*_{\phi(N)}$, computes the corresponding decryption exponent $d = e^{-1} \bmod \phi(N)$ and returns (N, e, d);*
- *The encryption function $f : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f(x) = x^e \bmod N$;*
- *The decryption function $f^{-1} : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f^{-1}(y) = y^d \bmod N$.*

In the following, we state our result in terms of the RSA primitive with a randomly chosen public exponent. The same results apply to the common practice of choosing a small public exponent. Actually, using Coppersmith's algorithm [7] as in [17] for OAEP [2], it would be possible to obtain tigther bounds for a small public exponent.

5.2 Partial-domain one-Wayness of RSA

The following lemma shows that the partial-domain one-wayness of RSA is equivalent to the one-wayness of RSA. This is a generalization of the result that appears in [9] for OAEP and in [4] for SAEP$^+$, wherein the size of the partial pre-image is greater than half the size of the modulus. The extension was announced in [9] and [4], even if the proper estimates were not worked out.

The technique goes as follows. Given $y = x^e \bmod N$, we must find x. We obtain the least significant bits of $x \cdot \alpha_i \bmod N$ for random integers $\alpha_i \in \mathbb{Z}_N$, by querying for the partial pre-image of $y_i = y \cdot (\alpha_i)^e \bmod N$. Finding x from the least significant bits of the $x \cdot \alpha_i \bmod N$ is a Hidden Number Problem modulo N. We use an algorithm similar to [5] to efficiently recover x.

Lemma 2. *Let $\mathcal{A}$ be an algorithm that on input y, outputs a q-set containing the k_1 most significant bits of $y^d \bmod N$, within time bound t, with probability ε, where $2^{k-1} \leq N < 2^k$, $k_1 \geq 64$ and $k/(k_1)^2 \leq 2^{-6}$. Then there exists an algorithm $\mathcal{B}$ that solves the RSA problem with success probability ε' within time bound t', where*

$$\begin{cases} \varepsilon' \geq \varepsilon \cdot (\varepsilon^{n-1} - 2^{-k/8}) \\ t' \leq n \cdot t + q^n \cdot \mathtt{poly}(k) \\ n = \left\lceil \dfrac{5k}{4k_1} \right\rceil \end{cases} . \tag{11}$$

Proof. See the full paper [8]. □

5.3 RSA-PSS-E and RSA-PSS-ES

The RSA-PSS-E encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ based on the PSS-R padding μ with parameters k, k_0, and k_1 is defined as follows:

- $\mathcal{K}$ generates a $(k+1)$-bit RSA modulus and exponents e and d. The public key is (N, e) and the private key is (N, d);
- $\mathcal{E}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random $r \in \{0,1\}^{k_0}$, the encryption algorithm outputs the ciphertext

$$c = (\mu(M, r))^e \bmod N$$

- $\mathcal{D}(c)$: the decryption algorithm recovers $x = c^d \bmod N$. It returns Reject if the most significant bit of x is not zero. It writes x as $0\|\omega\|s$ where ω is a k_1-bit string and s is a $k - k_1$ bit string. It writes $M\|r = G(\omega) \oplus s$. If $\omega = H(M||r)$, the algorithm returns M, otherwise it returns Reject.

The RSA-PSS-ES encryption and signature scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$ is defined as follows:

- $\mathcal{K}$, $\mathcal{E}(M, r)$ and $\mathcal{D}(c)$ are identical to RSA-PSS-E;
- $\mathcal{S}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random value $r \in \{0,1\}^{k_0}$, the signing algorithm computes the signature

$$\sigma = \mu(M, r)^d \bmod N$$

- $\mathcal{V}(\sigma)$: given the signature σ, the verification algorithm recovers $x = \sigma^e \bmod N$. It returns Reject if the most significant bit of x is not zero. It writes x as $0\|\omega\|s$ where ω is a k_1-bit string and s is a $(k - k_1)$-bit string. It writes $M\|r = G(\omega) \oplus s$. If $\omega = H(M||r)$, the algorithm accepts the signature and returns M, otherwise it returns Reject.

5.4 Security of RSA-PSS-E and RSA-PSS-ES

Combining Lemma 1 and Lemma 2, we obtain the following theorem which shows that the encryption scheme RSA-PSS-E is provably secure in the random oracle model, assuming that inverting RSA is hard.

Theorem 4. *Let $\mathcal{A}$ be a CCA2-adversary against the semantic security of the RSA-PSS-E scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, with advantage ε and running time t, making q_D, q_H and q_G queries to the decryption oracle and the hash function H and G, respectively. Provided that $k_1 \geq 64$ and $k/(k_1)^2 \leq 2^{-6}$, RSA can be inverted with probability ε' greater than*

$$\varepsilon' \geq \left(\varepsilon - q_H \cdot 2^{-k_0} - q_D 2^{-k_1}\right)^n - 2^{-k/8}$$

within time bound $t' \leq n \cdot t + (q_H + q_G)^n \cdot \mathtt{poly}(k)$, where $n = \lceil 5k/(4k_1) \rceil$.

We obtain a similar theorem for the semantic security of the RSA-PSS-ES encryption and signature scheme (from Lemma 2 and Lemma 3 in appendix A).

Theorem 5. *Let $\mathcal{A}$ be a CCA2-adversary against the semantic security of the RSA-PSS-ES scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$, with advantage ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle and the hash function H and G, respectively. Provided that $k_1 \geq 64$ and $k/(k_1)^2 \leq 2^{-6}$, RSA can be inverted with probability ε' greater than*

$$\varepsilon' \geq \left(\varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - (q_D + (q_H + q_{sig})^2) \cdot 2^{-k_1}\right)^n - 2^{-k/8}$$

within time bound $t' \leq n \cdot t + (q_H + q_G + q_{sig})^n \cdot \mathtt{poly}(k)$, where $n = \lceil 5k/(4k_1) \rceil$.

For the unforgeability of the RSA-PSS-ES encryption and signature scheme, we obtain a better security bound than the general result of Theorem 3, by relying upon the homomorphic properties of RSA. The proof of the following theorem is similar to the security proof of PSS in [3] and is given in the full version of this paper [8].

Theorem 6. *Let $\mathcal{A}$ be an adversary against the unforgeability of the PSS-ES scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash functions H and G, respectively. Then RSA can be inverted with probability ε' greater than*

$$\varepsilon' \geq \varepsilon - \left((q_H + q_{sig})^2 + q_D + 1\right) \cdot (2^{-k_0} + 2^{-k_1}) \tag{12}$$

within time bound $t' \leq t + (q_H + q_{sig}) \cdot \mathcal{O}(k^3)$.

Note that as for OAEP [9], the security proof for encrypting with PSS is far from being tight. This means that it does not provide a meaningful security result for a moderate size modulus (e.g., 1024 bits). For the security proof to be meaningful in practice, we recommend to take $k_1 \geq k/2$ and to use a larger modulus (e.g., 2048 bits).

6 Conclusion

In all existing PKIs different padding formats are used for encrypting and signing; moreover, it is recommended to use different keys for encrypting and signing. In this paper we have proved that the PSS padding scheme used in PKCS#1 v.2.1 [14] and IEEE P1363 [12] can be safely used for encryption as well. We have also proved that the same key pair can be safely used for both signature and encryption. The practical consequences of this are significant: besides halving the number of keys in security systems and simplifying their architecture, our observation allows resource-constrained devices such as smart cards to use the same code for implementing both signature and encryption.

Acknowledgements

We wish to thank Jacques Stern for pointing out an error in an earlier version of this paper, and the anonymous referees for their useful comments.

References

1. M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.
2. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Proceedings of Eurocrypt'94, LNCS vol. 950, Springer-Verlag, 1994, pp. 92–111.
3. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
4. D. Boneh, *Simplified OAEP for the RSA and Rabin functions*, Prooceedings of Crypto 2001, LNCS vol 2139, pp. 275-291, 2001.
5. D. Boneh and R. Venkatesan, *Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes*. Proceedings of Crypto '96, pp. 129-142, 1996.
6. R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, STOC' 98, ACM, 1998.
7. D. Coppersmith, *Finding a small root of a univariate modular equation*, in Eurocrypt '96, LNCS 1070.
8. J.S. Coron, M. Joye, D. Naccache and P. Paillier, *Universal padding schemes for RSA*. Full version of this paper. Cryptology ePrint Archive, `http://eprint.iacr.org/`.
9. E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, *RSA-OAEP is secure under the RSA assumption*, Proceedings of Crypto' 2001, LNCS vol. 2139, Springer-Verlag, 2001, pp. 260-274.
10. S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2), pp. 281-308, April 1988.
11. S. Haber and B. Pinkas, *Combining Public Key Cryptosystems*, Proceedings of the ACM Computer and Security Conference , November 2001.
12. IEEE P1363a, *Standard Specifications For Public Key Cryptography: Additional Techniques*, available at `http://www.manta.ieee.org/groups/1363/`.
13. J. Manger, *A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0*. Proceedings of Crypto 2001, LNCS 2139, pp. 230-238, 2001.
14. PKCS #1 v2.1, *RSA Cryptography Standard (draft)*, available at `http://www.rsasecurity.com/rsalabs/pkcs/`.
15. C. Rackoff and D. Simon, *Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack*. Advances in Cryptology, Crypto '91, pages 433-444, 1991.
16. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978.
17. V. Shoup, *OAEP reconsidered*, Proceedings of Crypto 2001, LNCS vol. 2139, pp 239-259, 2001.

A Proof of Theorem 2

The theorem follows from inequality (1) and the following lemma.

Lemma 3. *Let $\mathcal{A}$ be an adversary against the semantic security of PSS-ES, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash functions H and G, respectively. Then, the success probability $\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H+q_G+q_{sig}, t')$ is greater than*

$$\varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - q_D 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1}$$

where $t' \leq t + (q_H + q_{sig}) \cdot T_f$, and T_f denotes the time complexity of f.

Proof. The proof is very similar to the proof of Lemma 1. The top-level description of the reduction $\mathcal{B}$ is the same and the simulation of the decryption oracle is the same. However, oracles H and G are simulated differently. Instead of simulating H and G so that $\mu(M, r) = y$ is a random integer, we simulate H and G so that $\mu(M, r) = f(x)$ for a known random x, which allows to answer the signature query for M.

Simulation of oracles G and H and signing oracle. When receiving the query $M\|r$ to H, we generate a random $x \in \{0,1\}^k$ and compute $y = f(x)$. We denote $y = \omega\|s$. If ω never appeared before, we let $G(\omega) = s \oplus (M\|r)$ and return ω, otherwise we abort.

When receiving a query ω for G, if $G(\omega)$ has already been defined, we return $G(\omega)$, otherwise we return a random $(k - k_1)$-bit integer.

When we receive a signature query for M, we generate a random k_0-bit integer r. If $M\|r$ was queried to H before, we know ω, s, y and x such that

$$H(M\|r) = \omega \quad \text{and} \quad G(\omega) = s \oplus (M\|r) \quad \text{and} \quad y = f(x) = \omega\|s$$

so we return the corresponding signature x. If $M\|r$ was never queried before, we simulate an H-query for $M\|r$ as previously: we pick a random $x \in \{0,1\}^k$ and compute $y = f(x)$. We denote $y = \omega\|s$. If ω never appeared before, we let $H(M\|r) = \omega$, $G(\omega) = s \oplus (M\|r)$ and return the signature x, otherwise we abort.

Analysis. As in Lemma 1, we denote by AskG the event: "ω^* has been asked to G" and by AskH the event: "there exists M' such that $M'||r^*$ has been queried to H"; we denote by DBad the event: "a valid ciphertext has been rejected by our simulation of the decryption oracle D". Moreover, we denote by SBad the event: "the reduction aborts when answering an H-oracle query or a signature query". As previously, we have:

$$\Pr[\mathsf{AskH} | \neg \mathsf{AskG}] \leq (q_H + q_{sig}) \cdot 2^{-k_0}$$

and

$$\Pr[\mathsf{DBad}] \leq q_D \cdot 2^{-k_1} \quad .$$

When answering an H-oracle query or a signature query, the integer ω which is generated is uniformly distributed because f is a permutation. Moreover, at most $q_H + q_{sig}$ values of ω can appear during the reduction. Therefore the probability that the reduction aborts when answering an H-oracle query or a signature query is at most $(q_H + q_{sig}) \cdot 2^{-k_1}$, which gives

$$\Pr[\mathsf{SBad}] \leq (q_H + q_{sig})^2 \cdot 2^{-k_1} \ .$$

We denote by Bad the event

$$\mathsf{Bad} = \mathsf{AskG} \vee \mathsf{AskH} \vee \mathsf{DBad} \vee \mathsf{SBad}$$

Let S denote the event: "the adversary outputs the correct value for b, i.e. $b = b'$". Conditioned on $\neg\mathsf{Bad}$, our simulation of oracles G, H, D and of the signing oracle are independent of b, and therefore the adversary's view is independent of b. This gives

$$\Pr[\mathsf{S}|\neg\mathsf{Bad}] = \frac{1}{2} \tag{13}$$

Moreover, conditioned on $\neg\mathsf{Bad}$, the adversary's view is the same as when interacting with (perfect) random oracles, decryption oracle and signing oracle, which gives

$$\Pr[\mathsf{S} \wedge \neg\mathsf{Bad}] \geq \frac{1}{2} + \frac{\varepsilon}{2} - \Pr[\mathsf{Bad}] \tag{14}$$

which yields as in Lemma 1

$$\Pr[\mathsf{Bad}] \geq \varepsilon \tag{15}$$

and eventually

$$\Pr[\mathsf{AskG}] \geq \varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - q_D \cdot 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1} \ .$$

□

B Proof of Theorem 3

From $\mathcal{A}$ we construct an algorithm $\mathcal{B}$, which receives as input c and outputs η such that $c = f(\eta)$.

Top-level description of the reduction $\mathcal{B}$.

1. $\mathcal{B}$ is given a function f and $c = f(\eta)$, for a random integer η.
2. $\mathcal{B}$ selects uniformly at random an integer $j \in [1, q_H]$.
3. $\mathcal{B}$ runs $\mathcal{A}$ with f. It simulates the decryption oracle, the signing oracle and random oracles H and G as described below. $\mathcal{B}$ maintains a counter i for the i-th query $M_i \| r_i$ to H. The oracles H and G are simulated in such a way that if $i = j$ then $\mu(M_i \| r_i) = c$.
4. $\mathcal{B}$ receives from $\mathcal{A}$ a forgery σ. Letting M and r be the corresponding message and random, if $(M, r) = (M_j, r_j)$ then $f(\sigma) = \mu(M_j \| r_j) = c$ and $\mathcal{B}$ outputs σ.

Simulation of the oracles G, H, D and signing oracle. When receiving the i-th query $M_i\|r_i$ to H, we distinguish two cases: if $i \neq j$, we generate a random $x_i \in \{0,1\}^k$ and compute $y_i = f(x_i)$. If $i = j$, we let $y_i = c$. In both cases we denote $y_i = \omega_i\|s_i$. If ω_i never appeared before, we let $G(\omega_i) = s_i \oplus (M_i\|r_i)$ and return ω_i, otherwise we abort.

When receiving a query ω for G, if $G(\omega)$ has already been defined, we return $G(\omega)$, otherwise we return a random $(k - k_1)$-bit integer.

When we receive a signature query for M, we generate a random k_0-bit integer r. If $M\|r$ was queried to H before, we have $M\|r = M_i\|r_i$ for some i. If $i \neq j$, we have:

$$H(M_i\|r_i) = \omega_i, \quad G(\omega_i) = s_i \oplus (M_i\|r_i) \text{ and } y_i = \omega_i\|s_i = f(x_i)$$

so we return the corresponding signature x_i, otherwise we abort. If $M\|r$ was never queried before, we simulate an H-query for $M\|r$ as previously: we generate a random $x \in \{0,1\}^k$ and compute $y = f(x)$. We denote $y = \omega\|s$. If ω never appeared before, we let $H(M\|r) = \omega$ and $G(\omega) = s \oplus (M\|r)$ and return the signature x, otherwise we abort.

The simulation of the decryption oracle is identical to that of Lemma 1.

Analysis. Let σ be the forgery sent by the adversary. If ω was not queried to G, we simulate a query to G as previously. Let $\omega\|s = f(\sigma)$ and $M\|r = G(\omega) \oplus s$. If $M\|r$ was never queried to H, then $H(M\|r)$ is undefined because there was no signature query for M; the probability that $H(M\|r) = \omega$ is then 2^{-k_1}. Otherwise, let $(M, r) = (M_i, r_i)$ be the corresponding query to H. If $i = j$, then $\mu(M_j, r_j) = c = f(\sigma)$ and $\mathcal{B}$ succeeds in inverting f.

Conditioned on $i = j$, our simulation of H and the signing oracle are perfect, unless some ω appears twice, which happens with probability less than $(q_H + q_{sig})^2 \cdot 2^{-k_1}$. As in Lemma 1, our simulation of D fails with probability less than $q_D \cdot 2^{-k_1}$. Consequently, the reduction $\mathcal{B}$ succeeds with probability greater than:

$$\frac{1}{q_H} \cdot \left(\varepsilon - 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1} - q_D \cdot 2^{-k_1}\right)$$

which gives (10).

Efficient Computation of Full Lucas Sequences

[Published in *Electronics Letters* **32**(6):537–538, 1996.]

Marc Joye[1] and Jean-Jacques Quisquater[2]

[1] Dép. de Mathématique (AGEL), Université catholique de Louvain
Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium
joye@agel.ucl.ac.be
[2] Dép. d'Électricité (DICE), Université catholique de Louvain
Place de Levant 3, B-1348 Louvain-la-Neuve, Belgium
jjq@dice.ucl.ac.be

Abstract. Recently, Yen and Laih (1995) proposed an algorithm to compute LUC digital signatures quickly. This signature is based on a special type of the Lucas sequence, V_k. The authors generalise their method to any type of Lucas sequence, and extend it to the 'sister' Lucas sequence, U_k. As an application, the order of an elliptic curve over $GF(2^m)$ is computed quickly.

Keywords. Number theory, cryptography.

1 Basic Facts

In this section, we only include the minimal amount of background necessary to understand the Letter. For a systematic treatment, see [2, 3].

Let P and Q be two rational integers, and let α be a root of $x^2 - Px + Q = 0$ in the field $\mathbb{Q}(\sqrt{D})$, where $D = P^2 - 4Q$ is a non-square. Let β be the conjugate of α, *i.e.* $\beta = \bar{\alpha}$. The *Lucas sequences* $\{U_k\}_{k\geq 0}$ and $\{V_k\}_{k\geq 0}$ with parameters P and Q are given by

$$U_k(P,Q) = \frac{\alpha^k - \beta^k}{\alpha - \beta} \ , \tag{1}$$

$$V_k(P,Q) = \alpha^k + \beta^k \ . \tag{2}$$

It can easily be shown that the numbers U_i and V_i satisfy the following relations:

$$U_{i+j} = U_i V_j - Q^j U_{i-j} \ , \tag{3}$$

$$V_{i+j} = V_i V_j - Q^j V_{i-j} \ . \tag{4}$$

2 Fast Algorithm for Lucas Sequences

Assume you have to compute $U_k(P,Q)$ and $V_k(P,Q)$. If we use the binary expansion of k, it can be expressed as $k = K_0$, where

$$K_j = \sum_{i=j}^{n-1} k_i 2^{i-j}, \quad k_i \in \{0,1\} \text{ and } k_{n-1} = 1 \ .$$

Hence,

$$K_{j-1} = k_{j-1} + 2K_j = (K_j + k_{j-1}) + K_j \ . \tag{5}$$

Using Eqs. (3) and (4), we obtain

$$U_{K_{j-1}} = U_{(K_j+k_{j-1})}V_{K_j} - Q^{K_j}U_{k_{j-1}} \ , \tag{6}$$

$$V_{K_{j-1}} = V_{(K_j+k_{j-1})}V_{K_j} - Q^{K_j}V_{k_{j-1}} \ . \tag{7}$$

At iteration j, let $(l_j, h_j) = (K_j, K_j + 1)$. From Eq. (7), we see that both V_{l_j} and V_{h_j} are needed to compute $V_{l_{j-1}}$, and so to compute V_k. This is not the case for U_k, as we shall see in the following theorem.

Theorem 1. *If k is odd, then the computation of U_k does not require the computation of U_{l_j} $(j \geq 1)$.*

Proof. Since k is odd (i.e. $k_0 = 1$), $U_k(= U_{l_0}) = U_{h_1}V_{l_1} - Q^{l_1}$. Thus, only the value of U_{h_1} is needed. We only need to show that the value of $U_{h_{j-1}}$ can be derived from U_{h_j}. By Eq. (5) and depending on the value of k_{j-1}, we have the following cases:

- if $k_{j-1} = 0$, then $(l_{j-1}, h_{j-1}) = (2l_j, l_j + h_j)$;
- if $k_{j-1} = 1$, then $(l_{j-1}, h_{j-1}) = (l_j + h_j, 2h_j)$.

Hence, if $k_{j-1} = 0$, then $h_{j-1}(= h_j + l_j = 2l_j + 1)$ is odd and $U_{h_{j-1}} = U_{h_j}V_{l_j} - Q^{l_j}$; otherwise, $h_{j-1}(= 2h_j)$ is even and $U_{h_{j-1}} = U_{h_j}V_{h_j}$.

We now are ready to give the algorithm that we shall extend to the case where k is even.

Remarks: (i) The presented algorithm is a left-to-right scanning one. Similar to [1], it is also possible to develop a right-to-left scanning algorithm, but that requires more temporary memories.
(ii) An implementation with Pari-GP [4] is available at

`ftp://math.math.ucl.ac.be/pub/joye/pari/lucas2.gp`

```
Inputs:  k = 2^s Σ_{i=s}^{n-1} k_i 2^{i-s}, (k_s = 1)
         P, Q
Outputs: (U_k, V_k)

U_h = 1; V_l = 2; V_h = P; Q_l = 1; Q_h = 1;
for j from n − 1 to s + 1 by −1
    Q_l = Q_l * Q_h;
    if k[j] == 1 then
        Q_h = Q_l * Q;
        U_h = U_h * V_h;
        V_l = V_h * V_l − P * Q_l;
        V_h = V_h * V_h − 2 * Q_h
    else
        Q_h = Q_l;
        U_h = U_h * V_l − Q_l;
        V_h = V_h * V_l − P * Q_l;
        V_l = V_l * V_l − 2 * Q_l
    fi
endfor
Q_l = Q_l * Q_h; Q_h = Q_l * Q;
U_h = U_h * V_l − Q_l;
V_l = V_h * V_l − P * Q_l;
Q_l = Q_l * Q_h;
for j from 1 to s
    U_h = U_h * V_l;
    V_l = V_l * V_l − 2 * Q_l;
    Q_l = Q_l * Q_l;
endfor
{U_k(P, Q) = U_h; V_k(P, Q) = V_l}
```

Fig. 1. Algorithm to compute (U_k, V_k)

3 Performances

The worst case of the algorithm appears when $s = 0$. Assume $s = 0$; then the computation of U_k and V_k requires $\frac{11n}{2}$ multiplications. Furthermore, only five temporary memories (with the same length as the output) are needed.

If we only want the value of V_k, the algorithm is the same as that presented on figure 1, except that we do not care of the U_h's. Thus, the computation of V_k requires $\frac{9n}{2}$ multiplications in the worst case with only four temporary memories.

Remarks: (i) Some applications use Lucas sequences with parameter $Q = \pm 1$. In that case, the computation of U_k and V_k requires less than $3n$ multiplications with three temporary memories.
(ii) If we want U_k and V_k modulo a number, the computation can be improved using the technique of the common-multiplicand [5].

(iii) Moreover, owing to the high regularity of the algorithm, it can be parallelized.

4 Applications

Lucas sequences have numerous applications in number theory. For example, the divisibility properties of the U_k's (see [3, pp. 54–59]) allows to test the primality of a number N for which the factorization of $N+1$ is partially given. It is also possible to develop efficient (pseudo) primality tests [6].

In this Letter, we shall see a less-known application. In 1985, the theory of elliptic curves emerged for cryptographic purposes. Since many cryptographic protocols [7, 8] require the knowledge of the order of an elliptic curve over $GF(2^m)$, *i.e.* $\#E(GF(2^m))$, we shall see how it can be computed using Lucas sequences.

Let p be a prime and $q = p^r$. Consider the elliptic curve $E/GF(q)$ such that $E(GF(q))$ has order $\#E(GF(q)) = q + 1 - t$. Using Weil theorem, we have

$$\#E(GF(q^l)) = q^l + 1 - \alpha^l - \beta^l \ , \tag{8}$$

where α and β are given from the factorization of $1-tT+qT^2 = (1-\alpha T)(1-\beta T)$.

Let $E/GF(2^r)$ be the non-supersingular elliptic curve given by the Weierstraß equation

$$E : y^2 + xy = x^3 + a_2x^2 + a_6 \ .$$

Assume r is small, so $t = 2^r + 1 - \#E(GF(2^r))$ can be computed by exhaustion. Moreover, if m is a multiple of l, then the curve E can be viewed as an elliptic curve over $GF(2^m)$. Hence, if we put $l = m/r$, then by Eq. (8)

$$\#E(GF(2^m)) = 2^m + 1 - V_l(t, 2^r) \ ,$$

where $V_l(t, 2^r)$ is the l^{th} term of the Lucas sequence $\{V_k\}$ with parameters $P = t$ and $Q = 2^r$.

Acknowledgments

The authors are grateful to Daniel Bleichenbacher and to Richard Pinch for providing useful informations about Lucas-based cryptosystems.

References

1. YEN, S.-M., and LAIH, C.-S.: 'Fast algorithms for LUC digital signature computation', *IEE Proc.-Comput. Digit. Tech.*, 1995, **142**, (2), pp. 165–169
2. RIESEL, H.: 'Prime numbers and computers methods for factorization' *in* 'Progress in Mathematics' (Birkhäuser, 1985), Vol. 57
3. RIBENBOIM, P.: 'The little book of big primes' (Springer, 1991)
4. BATUT, C., BERNARDI, D., COHEN, H., and OLIVIER, M.: 'User's guide to PARI-GP', January 1995

5. YEN, S.-M., and LAIH, C.-S.: 'Common-multiplicand multiplication and its applications to public key cryptography', *Electron. Lett.*, 1993, **29**, (17), pp. 1583–1584
6. POMERANCE, C., SELFRIDGE, J.L., and WAGSTAFF, S.S., JR.: 'The pseudoprimes to $25 \cdot 10^9$', *Math. of Comp.*, 1980, **35**, (151), pp. 1003–1026
7. MENEZES, A.J.: 'Elliptic curve public key cryptosystems' (Kluwer Academic Publishers, 1993)
8. MENEZES, A., QU, M., and VANSTONE, S.: 'Draft of IEEE P1363, chapter 6', November 1995

Optimal Left-to-Right Binary Signed-Digit Recoding

[Published in *IEEE Transactions on Computers* **49**(7):740–748, 2000.]

Marc Joye[1] and Sung-Ming Yen[2]

[1] Gemplus Card International
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
marc.joye@gemplus.com
[2] Laboratory of Cryptography and Information Security (LCIS)
National Central University, Chung-Li, Taiwan 320, R.O.C.
yensm@csie.ncu.edu.tw

Abstract. This paper describes new methods for producing optimal binary signed-digit representations. This can be useful in the fast computation of exponentiations. Contrary to existing algorithms, the digits are scanned from left to right (i.e., from the most significant position to the least significant position). This may lead to better performances in both hardware and software.

Keywords. Computer arithmetic, converter, signed-digit representation, redundant number representation, SD2 left-to-right recoding, canonical/non-adjacent/minimum-weight form, exponentiation, elliptic curves, smart-cards, cryptography.

1 Introduction

Methodology using *signed-digit* (SD) representations, also called *redundant number* representations, for fast parallel arithmetic were considered in the late 1950's by Avizienis [1]. More recently, algorithms using signed-digit representations with the *digit set* $\{-1, 0, 1\}$ (in this paper, we call it the SD2 representation) accompanied with their applications to efficient methods for addition, multiplication, division, and their VLSI chip designs are presented in [2, 3, 4, 5, 6]. In general, after the computations in the SD2 domain, a *converter* is required to transform a number from its SD2 representation into the conventional binary representation. Such converters may be found in [7, Section 1.5], [8].

For many cryptosystems, (modular) *exponentiation* is one of the most time-consuming operations. Therefore, efficient algorithms to perform this operation are crucial in the performance of the resulting cryptographic protocols. Basically, when computing α^r, two types of exponentiations may be distinguished.

– The first type involves exponentiations with a fixed exponent r, such as in the RSA cryptosystem [9, 10]. The goal is then to quickly compute α^r for

randomly chosen α. This is usually achieved thanks to addition chains [11, Section 4.6.3]. The problem of finding the shortest addition chain was shown to be **NP**-hard by Downey, Leong and Sethi [12]; but good heuristics are known [13].
– In the second type of exponentiation, the base α is fixed and the exponent r varies. Examples include the ElGamal cryptosystem [14] and its numerous variations [15, Section 11.5]. In that case, good performances are obtained by the basic square-and-multiply technique (see Section 2). If larger amount of storage is available, this can be further improved via precomputations [16].

In this paper, we are mainly concerned with the second type of exponentiation. However, we note that our methods may lead to some advantages in the first type, too. Another application is when inverses can be virtually computed for free, as for elliptic curves [17]. The basic idea is to *recode* the exponent in a representation which has fewer nonzero digits, namely the SD2 representation. Already in 1951, this was successfully exploited by Booth to efficiently multiply two numbers [18]. An optimal version (in terms of the number of zero digits of the recoding) was later given by Reitwiesner [19]. In [20], Jedwab and Mitchell rediscovered Reitwiesner's algorithm and slightly generalized it by taking as input any SD2 representation of the exponent (instead of the binary representation). Our algorithms also produce optimal outputs but scan the digits of the exponent from left to right, i.e., from the most significant digit (MSD) to the least significant digit (LSD). This brings some advantages, especially in the hardware realization or for memory-constrained environments like smart-cards.

The rest of this paper is organized as follows. In Section 2, we review the square-and-multiply methods for fast exponentiation and extend them to exponents given in the SD2 representation. Section 3 presents Reitwiesner's algorithm. New exponent recoding algorithms are proposed in Section 4. Their hardware implementation is given in Section 5. Section 6 discusses further advantages of the proposed methods. Finally, we conclude in Section 7.

Notations

If $r = \sum_{i=0}^{m-1} r_i\, 2^i$ denotes the binary expansion of r, then we represent r as the vector $(r_{m-1}, \ldots, r_0)_2$. The bit-length of r is denoted by $|r|$. By abuse of notations, we do not make the distinction between the value of r and its representation and write $r = (r_{m-1}, \ldots, r_0)_2$. For signed-digit systems, we sometimes write $\bar{1}$ for -1. Moreover, if $r = \sum_{i=0}^{m} r'_i\, 2^i$ denotes the binary signed expansion of r (that is, $r'_i \in \{\bar{1}, 0, 1\}$), then we also abuse the notations and write $r = (r'_m, \ldots, r'_0)_{\text{SD2}}$.

Let S be a string. Then $\langle S \rangle^k$ means $S, S, \ldots, S$ (k times); for example, $(\langle 0, 1 \rangle^2, \bar{1})_{\text{SD2}}$ represents $(0, 1, 0, 1, \bar{1})_{\text{SD2}}$.

If t is a real number, then $\lfloor t \rfloor$ is the largest integer $\leq t$ and $\lceil t \rceil$ is the least integer $\geq t$.

Throughout this paper, the multiplicative notation is used. However, the described techniques also apply to additively written sets (e.g., additive group

of integers, points on an elliptic curve over a field). Exponentiation has then to be understood as multiplication.

2 Binary Algorithms for Fast Exponentiation

The most commonly used algorithms for computing α^r are the *binary methods* [11, Section 4.6.3]. The binary methods (also called *square-and-multiply methods*) scan the bits of exponent r either from right to left or from left to right (Fig. 1). At each step, a squaring is performed and depending on whether the scanned bit-value is equal to 1, a multiplication is also performed. Let $r = \sum_{i=0}^{m-1} r_i\, 2^i$ (with $r_{m-1} = 1$) be the binary expansion of r. The right-to-left (RL) algorithm is based on the observation that $\alpha^r = \left(\alpha^{2^0}\right)^{r_0} \left(\alpha^{2^1}\right)^{r_1} \cdots \left(\alpha^{2^{m-1}}\right)^{r_{m-1}}$, while the left-to-right (LR) algorithm follows from

$$\alpha^r = \left(\cdots \left(\left((\alpha^{r_{m-1}})^2\, \alpha^{r_{m-2}} \right)^2 \alpha^{r_{m-3}} \right)^2 \cdots \alpha^{r_1} \right)^2 \alpha^{r_0} \ .$$

```
INPUT: α, r = (r_{m-1}, ..., r_0)_2
OUTPUT: M = α^r
  M ← 1; S ← α
  for i from 0 to m − 1 do
    if (r_i = 1) then M ← M · S
    S ← S^2
  od
```

(a) Right-to-left (RL).

```
INPUT: α, r = (r_{m-1}, ..., r_0)_2
OUTPUT: M = α^r
  M ← 1
  for i from m − 1 down to 0 do
    M ← M^2
    if (r_i = 1) then M ← M · α
  od
```

(b) Left-to-right (LR).

Fig. 1. Square-and-multiply algorithms.

We remark that the LR algorithm (Fig. 1 (b)) requires 2 registers (for α and for M) and that the RL algorithm (Fig. 1 (a)) requires one more register (for S). However, we note that S can be used in place of α if the value of α is not needed thereafter. The RL algorithm presents the advantage to be parallelizable: one multiplier performs the multiplications $M \leftarrow M \cdot S$ and another one performs the squarings $S \leftarrow S^2$. However, if only one multiplier is available, the LR algorithm may be preferred because the multiplications are always done by the fixed value α, $M \leftarrow M \cdot \alpha$. So, if α has a special structure, these multiplications may be easier than multiplying two arbitrary numbers (see [15, Note 14.81] or [21, pp. 9–10] for examples of application).

Let $\omega(r)$ denote the *Hamming weight* of r (that is, the number of 1's in the binary representation of r). Both algorithms require $\omega(r) - 1$ multiplications and $m-1$ squarings (we do not count multiplications by 1, nor $1 \cdot 1$, nor the last squaring in Fig. 1 (a)) to compute α^r. It is well-known that $m-1$ is a lower bound for the number of squarings. However, the number of subsequent multiplications can

be further reduced by using a recoding algorithm [20, 22]. For example, to compute α^{15}, the LR algorithm will successively evaluate $\alpha, \alpha^2, \alpha^3, \alpha^6, \alpha^7, \alpha^{14}, \alpha^{15}$, that is, it performs 3 squarings and 3 multiplications. If the value of α^{-1} is supplied along with α (or if α^{-1} can cheaply be computed, as for elliptic curves [17]), then α^{15} is more quickly evaluated as $\alpha^{16} \cdot \alpha^{-1} = (((\alpha^2)^2)^2)^2 \cdot \alpha^{-1}$, which requires 4 squarings and 1 multiplication.

If we allow the digits of the exponent to be in $\{\bar{1}, 0, 1\}$, then the binary methods to compute α^r are easily modified as depicted in Fig. 2. Note that the SD2 representation of r may require an extra digit, r'_m, we refer to Section 3 for an explanation.

```
INPUT: α, r = (r'_m, ..., r'_0)_SD2
OUTPUT: M = α^r
  M ← 1; S ← α
  for i from 0 to m do
    if (r'_i = 1) then M ← M · S
    if (r'_i = 1̄) then M ← M · S^{-1}
    S ← S^2
  od
```

(a) Modified right-to-left (MRL).

```
INPUT: α, r = (r'_m, ..., r'_0)_SD2
OUTPUT: M = α^r
  M ← 1
  for i from m down to 0 do
    M ← M^2
    if (r'_i = 1) then M ← M · α
    if (r'_i = 1̄) then M ← M · α^{-1}
  od
```

(b) Modified left-to-right (MLR).

Fig. 2. Square-and-multiply-or-divide algorithms.

We see that the modified right-to-left (MRL) algorithm (Fig. 2 (a)) now works more differently and less efficiently. Indeed, when $r'_i = \bar{1}$, the inverse of S has to be computed (note that the value of S varies at each step). Most commonly used cryptosystems work in the multiplicative group of a finite field or ring. Inversion is then usually achieved via the extended Euclidean algorithm [15, Algorithm 2.142] or Fermat's Theorem [15, Fact 2.127] (see also [23] for a specialized implementation). This is a rather costly operation; therefore the benefits resulting from the reduced number of multiplications may be annihilated. The modified left-to-right (MLR) algorithm (Fig. 2 (b)) only needs the *fixed* value of α^{-1}, which can be precomputed. Consequently, in the sequel, we will only consider the MLR algorithm. We note, however, that the MRL algorithm may be useful when the inverse is available at no cost, as for elliptic curves or for the additive group of integers.

With the SD2 representation, the (minimal) Hamming weight $\omega(r)$ of r (i.e., the number of nonzero digits in the SD2 expansion of r) is equal to $(m+1)/3$, on average [24]. The computation of α^r can thus be performed with $\frac{4}{3}\,m + \mathrm{O}(1)$ multiplications plus squarings with the MLR algorithm, while the (standard) binary algorithms need $\frac{3}{2}\,m + \mathrm{O}(1)$ multiplications plus squarings (see Fig. 1), on average. Assuming that a squaring is approximatively as costly as a multiplication, then we can roughly expect a gain of $(\frac{3}{2} - \frac{4}{3})/\frac{3}{2} \approx 11.11\,\%$ over the (standard) binary methods.

The next section presents an algorithm to convert the exponent r from its binary representation into its SD2 representation. This algorithm, due to Reitwiesner, is optimal in the sense that it gives an SD2 output with *minimal* Hamming weight. Unfortunately, Reitwiesner's algorithm scans the bits of the exponent from *right to left*, while we have seen that only the modified *left-to-right* algorithm may bring some advantages. These *heterogeneous* modes of operation require to first recode the exponent r into its SD2 representation $(r'_m, \ldots, r'_0)_{\text{SD2}}$ and to temporarily store it for its latter usage in the MLR exponentiation algorithm. Noting that each digit in SD2 representation is encoded with 2 bits, *twice* the memory space taken by r is needed to store its SD2 representation. These shortcomings are alleviated in Section 4, where optimal left-to-right recoding algorithms are presented.

3 Reitwiesner's Method

In binary signed-digit notation[1] (i.e., using the digits $\{\bar{1}, 0, 1\}$), a number is not uniquely represented. Two representations $(a_\ell, a_{\ell-1}, \ldots, a_0)_{\text{SD2}}$ and $(b_\ell, b_{\ell-1}, \ldots, b_0)_{\text{SD2}}$ of a same number are *equivalent* if they have both the same length and the same Hamming weight; this equivalence will be denoted $(a_\ell, a_{\ell-1}, \ldots, a_0)_{\text{SD2}} \equiv (b_\ell, b_{\ell-1}, \ldots, b_0)_{\text{SD2}}$. A binary signed-digit representation is said to be *canonical* (or *sparse*) if no two adjacent digits are nonzero. For that reason, some authors sometimes call it the *nonadjacent form* (NAF) of a number [25].

The canonical recoding was studied by Reitwiesner [19]. He proved that this representation is unique (if the binary representation is viewed as padded with an initial 0). Following Hwang [7, pp. 150–151], Reitwiesner's method to convert a number $r = (r_{m-1}, \ldots, r_0)_2$ with $r_i \in \{0, 1\}$ into its canonical form $r = (r'_m, r'_{m-1}, \ldots, r'_0)_{\text{SD2}}$ with $r'_i \in \{\bar{1}, 0, 1\}$ is given by the algorithm depicted in Fig. 3. This is also known as Booth canonical recoding algorithm; however, Booth's method does not present the NAF property (see Footnote 4).

```
INPUT: (r_{m-1}, ..., r_0)_2
OUTPUT: (r'_m, r'_{m-1}, ..., r'_0)_SD2
  c_0 ← 0; r_{m+1} ← 0; r_m ← 0
  for i from 0 to m do
    c_{i+1} ← ⌊(c_i + r_i + r_{i+1})/2⌋
    r'_i ← c_i + r_i − 2c_{i+1}
  od
```

Fig. 3. Reitwiesner's canonical recoding algorithm.

Reitwiesner's algorithm is very efficient. It can be done by using the following look-up table ('X' stands for 0 or 1, that is, the output is independent of this value).

[1] Also called ternary balanced notation [11, p. 190].

Table 1. Right-to-left SD exponent recoding.

c_i	r_i	r_{i+1}	c_{i+1}	r'_i
0	0	X	0	0
0	1	0	0	1
0	1	1	1	$\bar{1}$
1	0	0	0	1
1	0	1	1	$\bar{1}$
1	1	X	1	0

At first glance, it is not so obvious that this algorithm effectively yields the canonical representation of a number. However, if we closely observe how it works, we see that this algorithm comes down to subtract r from $3r$ (with the additional rule $0 - 1 = \bar{1}$) and then to discard the last (i.e., least significant) 0 [26].

$$\begin{array}{rr} 2r = & (r_{m-1},\ r_{m-2}\ ,\ r_{m-3},\ \ldots,\ r_1,\ r_0,\ 0\ )_2 \\ +\,r = & (r_{m-1},\ r_{m-2},\ \ldots,\ r_2,\ r_1,\ r_0)_2 \\ \hline 3r = & (s_m,\ s_{m-1}\ ,\ s_{m-2}\ ,\ s_{m-3},\ \ldots,\ s_1,\ s_0,\ r_0)_2 \\ -\,r = & (r_{m-1},\ r_{m-2},\ \ldots,\ r_2,\ r_1,\ r_0)_2 \\ \hline 2r = & (r'_m,\ r'_{m-1}\ ,\ r'_{m-2}\ ,\ r'_{m-3},\ \ldots,\ r'_1,\ r'_0,\ 0\ )_{\text{SD2}} \end{array}$$

Fig. 4. A simple explanation of Reitwiesner's method.

Indeed, if $r_0 + \sum_{i=0}^{m} s_i\, 2^{i+1}$ denotes the binary expansion of $3r$, then the conventional pencil-and-paper method to add nonnegative integers [11, p. 251] gives $s_i = (c_i + r_i + r_{i+1}) \bmod 2 = c_i + r_i + r_{i+1} - 2\lfloor (c_i + r_i + r_{i+1})/2 \rfloor$ where c_i is the carry-in. Moreover, since the carry-out c_{i+1} is equal to 1 if and only if there are two or three 1's among c_i, r_i and r_{i+1}, we can write $c_{i+1} = \lfloor (c_i + r_i + r_{i+1})/2 \rfloor$. Hence, $s_i = c_i + r_i + r_{i+1} - 2c_{i+1}$ and thus $r'_i = s_i - r_{i+1} = c_i + r_i - 2c_{i+1}$.

To see that the output is sparse, it suffices to remark that the algorithm scans the bits from right to left and replaces a consecutive block of several 1's by a block of 0's and $\bar{1}$ according to $(\langle 1 \rangle^a)_2 \mapsto (1, \langle 0 \rangle^{a-1}, \bar{1})_{\text{SD2}}$.[2] Furthermore, if two blocks of 1's are separated by an isolated 0, the algorithm implicitly uses the fact that $(\bar{1}, 1)_{\text{SD2}} \equiv (0, \bar{1})_{\text{SD2}}$. For example, $(\langle 1 \rangle^a, 0, \langle 1 \rangle^b)_2$ is replaced by $(1, \langle 0 \rangle^a, \bar{1}, \langle 0 \rangle^{b-1}, \bar{1})_{\text{SD2}}$—and not by $(1, \langle 0 \rangle^{a-1}, \bar{1}, 1, \langle 0 \rangle^{b-1}, \bar{1})_{\text{SD2}}$. A more formal (but less intuitive) proof of the sparseness property is given in the next lemma.

Lemma 1. $r'_i \cdot r'_{i+1} = 0$ *for all* $0 \le i \le m - 1$.

Proof. Suppose $r'_i \neq 0$. Since $r'_i = c_i + r_i - 2c_{i+1}$, we must have $c_i + r_i = 1$, whence $c_{i+1} = \lfloor (1 + r_{i+1})/2 \rfloor = r_{i+1}$ and thus $r'_{i+1} = 2(r_{i+1} - c_{i+2}) = 0$. □

[2] This is the transformation initially proposed by Booth [18].

The main advantage of Reitwiesner's algorithm is that, in some sense, it is optimal. Indeed, Reitwiesner [19] proved that:

Proposition 1. *Among the* SD2 *representations, the canonical representation has minimal Hamming weight.* □

Although, this does not rule out the existence of other minimal representations. For example, $(1,0,1,1)_{\text{SD2}}$ and $(1,0,\bar{1},0,\bar{1})_{\text{SD2}}$ are both minimal representations for 11.

The general case was later addressed by Clark and Liang [26]. They present a minimal representation for any signed-radix b. In that case, Arno and Wheeler [24] proved that the average proportion of nonzero digits is equal to $(b-1)/(b+1)$. This has to be compared with the average proportion $(b-1)/b$ of nonzero digits in the standard radix b representation. So, we see that exponent recoding is mostly interesting for binary signed-digit representation ($b = 2$) because the savings rapidly go down.

4 Proposed Methods

In Section 2, we pointed out that a left-to-right recoding algorithm might be desirable for fast exponentiation. Designing such an algorithm is not as straightforward as it appears and is even considered as a hard problem by some authors [27].

Our first algorithm is an adaptation of Reitwiesner's algorithm. As in the right-to-left algorithm, it also presents the NAF property. Unfortunately, look-up tables cannot be used. Our second algorithm does not have the NAF property but is *equally* efficient as Reitwiesner's algorithm. Moreover, it enables the use of a look-up table.

4.1 A simple left-to-right recoding algorithm

The interpretation of the NAF given in the previous section suggests a simple way to construct a left-to-right recoding algorithm. Let $r = (r_{m-1}, \ldots, r_0)_2$ and $3r = (s_m, \ldots, s_0, r_0)_2$, then the NAF for r is $(r'_m, r'_{m-1}, \ldots, r'_0)_{\text{SD2}}$ where $r'_i = s_i - r_{i+1}$. So, if we have at our disposal an algorithm to add $r = (r_{m-1}, \ldots, r_0)_2$ and $2r = (r_{m-1}, \ldots, r_0, 0)_2$ from *left to right*, then we can compute $3r$, subtract r (in SD2 representation), discard the last 0 and obtain the canonical representation of r. Fortunately, such algorithms exist. A left-to-right addition algorithm is presented in Fig. 5 (see [11, Exercise 4.3.1.6]).

Taking as input $u = (0, r_{m-1}, \ldots, r_1)_2$ and $v = (r_{m-1}, r_{m-2}, \ldots, r_0)_2$, we get $w = (s_m, \ldots, s_0)_2$; we have now to subtract $(r_{m-1}, \ldots, r_1)_2$ (with the rule $0 - 1 = \bar{1}$). However, some technical difficulties occur: when we enter in the `while loop` (Lines 5–7 and 12–14 in Fig. 5), the algorithm may output *several* s_j's (with $j > i$) wherefrom, for each s_j, we have to subtract r_{j+1} in order to obtain the corresponding r'_j. This means that the r_{j+1}'s with $j > i$ (i.e.,

```
INPUT: u = (u_{m-1}, ..., u_0)_2 and v = (v_{m-1}, ..., v_0)_2
OUTPUT: u + v = (w_m, w_{m-1}, ..., w_0)_2
  j ← m
  for i from m − 1 down to 0 do
    if (u_i = v_i) then
      w_j ← v_i
      while (j > i + 1) do
        j ← j − 1; w_j ← 1 − v_i
      od
      j ← j − 1
    fi
  od
  w_j ← 0
  while (j > 0) do
    j ← j − 1; w_j ← 1
  od
```

Fig. 5. Left-to-right addition algorithm.

$r_{i+2}, r_{i+3}, \ldots$) must be available. On the other hand, we remark that at step $i = I$, variable j is only decremented when $r_{I+1} = r_I$. So, if J denotes the value of j before entering in the `while loop`, we see that this loop is only executed after a block $(r_{J+1}, r_J, \ldots, r_{I+1}, r_I)$ either of the form

$$\text{(B1)} \qquad (0, \langle 0,1\rangle^{(J-i)/2}, 1) \quad \text{or} \quad (0, \langle 0,1\rangle^{(J-i-1)/2}, 0, 0) \ ,$$

or of the form

$$\text{(B2)} \qquad (1, \langle 1,0\rangle^{(J-i)/2}, 0) \quad \text{or} \quad (1, \langle 1,0\rangle^{(J-i-1)/2}, 1, 1) \ .$$

We therefore introduce an additional variable b to distinguish between the two cases. Because of the alternation of 0 and 1 inside a block, we do not have to know the value of the r_{j+1}'s inside the `while loop`; only the value of the two first consecutive equal bits (i.e., r_{J+1} and r_J) is necessary: we use the variable b to keep track of this value, that is, before entering the loop, b contains the value of r_J ($b = 0$ in Case (B1) and $b = 1$ in Case (B2)). Next, inside the `while loop`, we alternately subtract 0 or 1, starting with 0 or 1 depending on the value of b. Putting all together, we finally our first algorithm (Fig. 6).

While this algorithm yields the canonical representation, it is not fully satisfying. It looks quite cumbersome compared to the original Reitwiesner's algorithm (see Fig. 3). The next paragraph considers another minimal representation which leads to a very elegant left-to-right recoding algorithm.

4.2 Minimum-weight left-to-right recoding algorithm

The main difficulty in the previous algorithm comes from the fact that it can only "decide" what the output will be after two consecutive equal bits. In other words,

```
INPUT: (r_{m-1}, ..., r_0)_2
OUTPUT: (r'_m, r'_{m-1}, ..., r'_0)_SD2
  j ← m; b ← 0; r_m ← 0
  for i from m − 1 down to 0 do
    if (r_{i+1} = r_i) then
      r'_j ← r_i − b
      while (j > i + 1) do
        j ← j − 1; r'_j ← 1 − r_i − b
        b ← 1 − b
      od
      b ← r_i; j ← j − 1
    fi
  od
  r'_j ← −b
  while (j > 0) do
    j ← j − 1; r'_j ← 1 − b
    b ← 1 − b
  od
```

Fig. 6. Canonical left-to-right recoding algorithm.

when entering in an alternate block of $\langle 0,1\rangle^k$ (or $\langle 1,0\rangle^k$), the algorithm must know *a priori* if this block will end with two consecutive bits equal to 0 or 1. The next lemma enables to give a *minimal* (albeit not sparse) output whatever the ending bits of an alternate block. The `while loop`'s in the canonical algorithm (Fig. 6) can therefore be removed.

Lemma 2. *In* SD2 *representation, we have the following equivalences:*

(a) $(\langle 0,1\rangle^k, 1)_{\text{SD2}} \equiv (1, \langle 0,\bar{1}\rangle^k)_{\text{SD2}}$*;*
(b) $(\langle 0,\bar{1}\rangle^k, \bar{1})_{\text{SD2}} \equiv (\bar{1}, \langle 0,1\rangle^k)_{\text{SD2}}$.

Proof. Let $S_k = \sum_{i=1}^{k} 2^{2(i-1)} = (2^{2k}-1)/3$. In signed-digit representation, $(\langle 0,1\rangle^k, 1)_{\text{SD2}}$ represents the number $N = 1 + \sum_{i=1}^{k} 2^{2(i-1)+1}$. However since $N = 1 + 2S_k = (3S_k + 1) - S_k = 2^{2k} - \sum_{i=1}^{k} 2^{2(i-1)}$, it may also be represented as $(1, \langle 0,\bar{1}\rangle^k)_{\text{SD2}}$. Noting that $(\langle 0,\bar{1}\rangle^k, \bar{1})_{\text{SD2}}$ and $(\bar{1}, \langle 0,1\rangle^k)_{\text{SD2}}$ are both representations of $-N$, the second equivalence follows from the first one. □

Elementary blocks [This paragraph explains how the recoding algorithm was found. The reader uniquely interested by the algorithm itself may skip it.]

The exponent r to be recoded may be viewed as a succession of *elementary blocks* $(r_{J+1}, r_J, \ldots, r_{I+1}, r_I)$ with $J > I$, $r_{J+1} = r_J$ and $r_{I+1} = r_I$, and whose form is given by one of the four following possibilities.

(E1) $(0, \langle 0,1\rangle^{(J-I)/2}, 1)$,
(E2) $(0, \langle 0,1\rangle^{(J-I-1)/2}, 0, 0)$,

(E3) $(1, \langle 1, 0\rangle^{(J-I)/2}, 0)$,
(E4) $(1, \langle 1, 0\rangle^{(J-I-1)/2}, 1, 1)$.

For each of these forms, our left-to-right canonical recoding algorithm (Fig. 6) will respectively output the corresponding block $(r'_J, \ldots, r'_{I+1})$ given by

(E1') $(1, \langle 0, \bar{1}\rangle^{(J-I-2)/2}, 0)$,
(E2') $(0, \langle 1, 0\rangle^{(J-I-1)/2})$,
(E3') $(\bar{1}, \langle 0, 1\rangle^{(J-I-2)/2}, 0)$,
(E4') $(0, \langle \bar{1}, 0\rangle^{(J-I-1)/2})$.

Now, using Lemma 2, we may respectively replace these outputs by

(E1*) $(\langle 0, 1\rangle^{(J-I-2)/2}, 1, 0)$,
(E2*) $(\langle 0, 1\rangle^{(J-I-1)/2}, 0)$, (= E2')
(E3*) $(\langle 0, \bar{1}\rangle^{(J-I-2)/2}, \bar{1}, 0)$,
(E4*) $(\langle 0, \bar{1}\rangle^{(J-I-1)/2}, 0)$. (= E4')

Example 1. Suppose that $r = (1, 1, 1, 0, 1, 0, 0, 1)_2$. By adding artificial beginning and ending 0's, we decompose r into 4 elementary blocks. We then apply our transformation to each elementary block to finally obtain $r = (1, 0, 0, 0, \bar{1}, \bar{1}, 0, 0, 1)_{\text{SD2}}$.

$$
\begin{array}{rl}
 & r = (\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ .\ 0\ 0\ )_2 \\
\text{(E1)} & \quad 0\ 0\ 1\ 1 \\
\text{(E4)} & \quad\ \ 1\ 1\ 1 \\
\text{(E3)} & \quad\ \ \ 1\ 1\ 0\ 1\ 0\ 0 \\
\text{(E2)} & \quad\ \ \ \ \ \ \ \ 0\ 0\ 1\ .\ 0\ 0 \\
\text{(E1*)} & \quad\ 1\ 0 \\
\text{(E4*)} & \quad\ \ \ 0 \\
\text{(E3*)} & \quad\ \ \ \ 0\ \bar{1}\ \bar{1}\ 0 \\
\text{(E2*)} & \quad\ \ \ \ \ \ \ \ \ \ 0\ 1\ .\ 0 \\
 & \rightarrow r = (\quad 1\ 0\ 0\ 0\ \bar{1}\ \bar{1}\ 0\ 0\ 1\ .\ 0\quad)_{\text{SD2}}
\end{array}
$$

Although not sparse, the resulting representation has the same Hamming weight as the canonical representation $(1, 0, 0, \bar{1}, 0, 1, 0, 0, 1)_{\text{SD2}}$. Both representations being equivalent, $(1, 0, 0, 0, \bar{1}, \bar{1}, 0, 0, 1)_{\text{SD2}}$ is thus a *minimal* representation for r. $\Diamond$

Consider an elementary block $(r_{J+1}, r_J, \ldots, r_{I+1}, r_I)$ and the corresponding signed-digit block $(r'_J, \ldots, r'_{I+1})$. Let b_i denote the value of variable b when bits r_i and r_{i-1} are scanned, namely $b_i = 0$ or 1 depending on whether r_i and r_{i-1} belong to an elementary block beginning with two zeros or two 1's. Suppose that $r_{J+1} = r_J = 0$. Then we have $b_i = 0$ for all $J + 1 \geq i \geq I + 2$, and $b_{I+1} = 0$ if $r_{I+1} = r_I = 0$ or $b_{I+1} = 1$ if $r_{I+1} = r_I = 1$. Similarly, if $r_{J+1} = r_J = 1$, then $b_i = 1$ for all $J + 1 \geq i \geq I + 2$, and $b_{I+1} = 0$ if $r_{I+1} = r_I = 0$ or $b_{I+1} = 1$ if $r_{I+1} = r_I = 1$. Because of the alternation of 0 and 1 inside an elementary block, we have $r_i + r_{i-1} = 1$ for all $J \geq i \geq I + 2$. Hence, if we set

$$b_i = \lfloor (b_{i+1} + r_i + r_{i-1})/2 \rfloor \ , \tag{1}$$

we have $b_i = \lfloor (b_{i+1} + 1)/2 \rfloor = b_{i+1}$ for all $J \geq i \geq I + 2$ and thus by induction $b_i = b_{J+1}$ for all $J \geq i \geq I + 2$ as required. Moreover, since $r_{I+1} = r_I$, we also have $b_{I+1} = \lfloor (b_{I+2} + 2r_{I+1} + r_I)/2 \rfloor = r_{I+1}$ as required. Finally, we remark that if $b_i = b_{i-1} = 0$ then $r'_i = r_i$, and that if $b_i = b_{i-1} = 1$ then $r'_i = r_i - 1$. Consequently, when $b_i = b_{i-1}$, we can write $r'_i = r_i - b_i$. If $b_i \neq b_{i-1}$, then (1) if $b_i = 1$ and $b_{i-1} = 0$ then $r'_i = \bar{1}$; (2) if $b_i = 0$ and $b_{i-1} = 1$ then $r'_i = 1$. So, when $b_i \neq b_{i-1}$, we can write $r'_i = b_{i-1} - b_i$. Noting that if $b_i \neq b_{i-1}$ then $b_i = r_i$, we can see that

$$r'_i = (r_i - b_i) + (b_{i-1} - b_i) = r_i - 2b_i + b_{i-1} \tag{2}$$

is a valid expression for r'_i.

Our second algorithm From the simple formulations for b_i and r'_i (see Eqs. (1) and (2)), we obtain an elegant and efficient left-to-right exponent recoding algorithm.

```
INPUT: (r_{m-1}, ..., r_0)_2
OUTPUT: (r'_m, r'_{m-1}, ..., r'_0)_SD2
  b_m ← 0; r_m ← 0; r_{-1} ← 0; r_{-2} ← 0
  for i from m down to 0 do
    b_{i-1} ← ⌊(b_i + r_{i-1} + r_{i-2})/2⌋
    r'_i ← −2b_i + r_i + b_{i-1}
  od
```

Fig. 7. Minimum-weight left-to-right recoding algorithm.

Similarly as Reitwiesner's algorithm, our second algorithm can be performed still more efficiently thanks to table look-up. The corresponding look-up table is given hereafter (as aforementioned 'X' stands for 0 or 1).

Table 2. Left-to-right SD exponent recoding.

b_i	r_i	r_{i-1}	r_{i-2}	b_{i-1}	r'_i
0	0	0	X	0	0
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	X	0	1
1	0	1	X	1	$\bar{1}$
1	1	0	0	0	$\bar{1}$
1	1	0	1	1	0
1	1	1	X	1	0

Note that entries $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 1, 1, \mathrm{X})$ and $(1, 0, 0, \mathrm{X})$ are missing; see the discussion before Lemma 3 (next paragraph) for an explanation.

Main theorem From its construction, the proposed algorithm (Fig. 7) produces an SD2 representation with minimal Hamming weight. However, for completeness, we now give a formal proof of this assertion.

We first consider three auxiliary lemmas. Lemma 3 implies that the cases $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 1, 1, \mathrm{X})$ and $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 0, 0, \mathrm{X})$ never occur (see Table 2). Lemma 4 shows that if $b_i = r_i$ then the output is sparse. Finally, Lemma 5 proves that our algorithm effectively yields an SD2 representation.

Lemma 3. *If $r_i = r_{i-1}$ then $b_i = r_i$.*

Proof. Straightforward because $r_i = r_{i-1}$ yields $b_i = \lfloor (b_{i+1} + r_i + r_{i-1})/2 \rfloor = \lfloor b_{i+1}/2 + r_i \rfloor = r_i$. □

Lemma 4. *If $b_i = r_i$ then $r'_i \cdot r'_{i-1} = 0$.*

Proof. Suppose first that $r_{i-1} + r_{i-2} = 2(1 - b_i)$. Then $r_{i-1} = r_{i-2} = 1 - b_i$. So, by Lemma 3, $b_{i-1} = 1 - b_i$ and thus $b_{i-2} = \lfloor (b_{i-1} + r_{i-2} + r_{i-3})/2 \rfloor = \lfloor (1 - b_i) + r_{i-3}/2 \rfloor = 1 - b_i$. Hence, $r'_{i-1} = -2b_{i-1} + r_{i-1} + b_{i-2} = 0$ and $r'_i \cdot r'_{i-1} = 0$. If $r_{i-1} + r_{i-2} \neq 2(1 - b_i)$, then $r_{i-1} + r_{i-2} \leq 1$ when $b_i = 0$ and $r_{i-1} + r_{i-2} \geq 1$ when $b_i = 1$. So, $b_{i-1} = \lfloor (b_i + r_{i-1} + r_{i-2})/2 \rfloor = 0$ when $b_i = 0$ and $b_{i-1} = 1$ when $b_i = 1$; or equivalently, $b_{i-1} = b_i$. Therefore, since $b_i = r_i$, we have $r'_i = -2b_i + r_i + b_{i-1} = 0$ and $r'_i \cdot r'_{i-1} = 0$. □

Lemma 5. *Let $r = (r_{m-1}, \ldots, r_0)_2$ be the binary representation of r. Then the output $(r'_m, \ldots, r'_0)_{\mathrm{SD2}}$ given by the minimum-weight left-to-right recoding algorithm is an SD2 representation of r.*

Proof. From the look-up table (which enumerates all the possible cases), we have $r'_i \in \{\bar{1}, 0, 1\}$. Hence, it remains to prove that $\sum_{i=0}^{m} r'_i 2^i = r$. The algorithm gives

$$\begin{aligned}\sum_{i=0}^{m} r'_i 2^i &= \sum_{i=0}^{m} (-2b_i + r_i + b_{i-1}) 2^i \\ &= \sum_{i=0}^{m} r_i 2^i + \sum_{i=0}^{m} (b_{i-1} 2^i - b_i 2^{i+1}) \\ &= r_m 2^m + r + (b_{-1} - b_m 2^{m+1}) = r\end{aligned}$$

since $r_m = b_m = 0$ (by definition) and $b_{-1} = 0$ (because $r_{-1} = r_{-2} = 0$). □

Theorem 1. *As Reitwiesner's algorithm, our left-to-right recoding algorithm produces an SD2 representation with minimal Hamming weight.*

Proof. Proposition 1 says that a sparse method gives a minimal output. So, from Lemmas 3 and 4, it remains to prove that the following inputs lead to optimal outputs.

1. $\boxed{(b_i, r_i, r_{i-1}, r_{i-2}) = (0,1,0,0)}$

   Hence, $(b_{i-1}, r_{i-1}, r_{i-2}, r_{i-3}) = (0,0,0,r_{i-3})$. So, again from Table 2, we have $r'_{i-1} = 0$ and thus $r'_i \cdot r'_{i-1} = 0$.

2. $\boxed{(b_i, r_i, r_{i-1}, r_{i-2}) = (0,1,0,1)}$

   Then we have $(b_{i-1}, r_{i-1}, r_{i-2}, r_{i-3}) = (0,0,1,r_{i-3})$ and $r'_i = 1$.
   (a) If $r_{i-3} = 0$ then, from Table 2, $r'_{i-1} = 0$ and thus $r'_i \cdot r'_{i-1} = 0$.
   (b) Otherwise if $r_{i-3} = 1$ then $b_{i-2} = 1$ and $r'_{i-1} = 1$. Furthermore, since $b_{i-1} = r_{i-1}$, it follows from Lemma 4 that $r'_{i-2} = 0$. We now examine the outputs on the left. Let $k \geq 1$ be the least integer such that $r_{i+2k} = 0$. Hence $r_{i+2j} = 1$ for all $0 \leq j \leq k-1$. Define $\vec{Y}_j = (b_{i+2j}, r_{i+2j}, r_{i+2j-1})$. We can prove by induction that

   $$\vec{Y}_j = (b_{i+2j}, r_{i+2j}, r_{i+2j-1}) = (0,1,0) \qquad (*)$$

   for all $0 \leq j \leq k-1$. If $j = 0$ then $\vec{Y}_0 = (0,1,0)$. Suppose now that $\vec{Y}_n = (b_{i+2n}, r_{i+2n}, r_{i+2n-1}) = (0,1,0)$ for some $0 \leq n \leq k-2$. We must then have $b_{i+2n+1} = 0$ because $b_{i+2n+1} = 1$ implies $b_{i+2n} = 1$, a contradiction. This, in turn, implies $r_{i+2n+1} = 0$ (otherwise, by Lemma 3, we would get $b_{i+2n+1} = r_{i+2n+1} = 1$). We also have $r_{i+2n+2} = 1$ because $r_{i+2j} = 1$ for all $0 \leq j \leq k-1$. Finally, since $b_{i+2n+1} = 0$, we must have $b_{i+2n+2} = 0$. Consequently, $\vec{Y}_{n+1} = (b_{i+2n+2}, r_{i+2n+2}, r_{i+2n+1}) = (0,1,0)$ and Eq. (*) is proven.
   Moreover, from Eq. (*), we have $b_{i+2j-1} = 0$ for all $0 \leq j \leq k-1$. Similarly, from $\vec{Y}_{k-1}$, we can prove that $b_{i+2k} = b_{i+2k-1} = 0$ and $r_{i+2k-1} = 0$. In short, when $r_{i-3} = 1$, we have:

r_{i+3}	r_{i+2}	r_{i+1}	r_i	r_{i-1}	r_{i-2}	r'_{i+3}	r'_{i+2}	r'_{i+1}	r'_i	r'_{i-1}	r'_{i-2}
·	·	0	1	0	1	·	·	0	1	1	0

   Since $(r_{i+2k}, \ldots, r_{i-1}, r_{i-2}) = (0, \langle 0,1 \rangle^{k+1})$ and $b_{i+j} = 0$ for all $0 \leq j \leq 2k$, we obtain $(r'_{i+2k}, \ldots, r'_i) = (0, \langle 0,1 \rangle^k)$. Moreover, $r'_{i-1} = 1$ and $r'_{i-2} = 0$. Therefore, the output is $(r'_{i+2k}, \ldots, r'_{i-1}, r'_{i-2}) = (0, \langle 0,1 \rangle^k, 1, 0) \equiv_{\text{SD2}} (0,1, \langle 0, \bar{1} \rangle^k, 0)$ by Lemma 2. Since the latter is sparse (note the beginning 0 and the ending 0), the output $(r'_{i+2k}, \ldots, r'_{i-1}, r'_{i-2})$ is optimal by Proposition 1.

3. $\boxed{(b_i, r_i, r_{i-1}, r_{i-2}) = (1,0,1,0)}$

   In this case, we have $(b_{i-1}, r_{i-1}, r_{i-2}, r_{i-3}) = (1,1,0,r_{i-3})$ and $r'_i = \bar{1}$.
   (a) If $r_{i-3} = 1$ then $r'_{i-1} = 0$ and thus $r'_i \cdot r'_{i-1} = 0$.
   (b) If $r_{i-3} = 0$, analogously to Case (2.b), we obtain:

r_{i+3}	r_{i+2}	r_{i+1}	r_i	r_{i-1}	r_{i-2}	r'_{i+3}	r'_{i+2}	r'_{i+1}	r'_i	r'_{i-1}	r'_{i-2}
·	·	1	0	1	0	·	·	0	$\bar{1}$	$\bar{1}$	0

Using similar arguments to those used in Case (2.b), we can prove that the output must be of the form $(r'_{i+2k}, \ldots, r'_{i-1}, r'_{i-2}) = (0, \langle 0, \bar{1}\rangle^k, \bar{1}, 0)$ for some $k \geq 1$. From Lemma 2, we see that this output is optimal since equivalent to the sparse representation $(0, \bar{1}, \langle 0, 1\rangle^k, 0)$.

4. $\boxed{(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 0, 1, 1)}$

   Similarly to Case (1.), $(b_{i-1}, r_{i-1}, r_{i-2}, r_{i-3}) = (1, 1, 1, r_{i-3})$; so $r'_{i-1} = 0$ and thus $r'_i \cdot r'_{i-1} = 0$.

This concludes the proof. □

5 Hardware Implementation

Evidently, the proposed minimum-weight signed-digit converter (Fig. 7) is much more regular and simpler for hardware implementation. To implement the transformation algorithm, we encode r'_i ($r'_i \in \{0, 1, \bar{1}\}$) as

$$\{0 \triangleq (\mathrm{X}, 0)_2;\ 1 \triangleq (0, 1)_2;\ -1 \triangleq (1, 1)_2\},$$

and we denote r'_i by a two-bit representation $(r'_{i,H}, r'_{i,L})_2$.

After some logic manipulations and minimizations, the following Boolean equations for b_{i-1}, $r'_{i,H}$ and $r'_{i,L}$ can be obtained. The outputs (b_{i-1}, r'_i) corresponding to the missing entries $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 1, 1, \mathrm{X})$ and $(1, 0, 0, \mathrm{X})$ of Table 2 were optimally assigned to $(0, 1)$ and $(1, \bar{1})$, respectively.

$$\begin{cases} b_{i-1} = \overline{b_i} \cdot r_{i-1} \cdot r_{i-2} + b_i \cdot r_{i-1} + b_i \cdot r_{i-2} \\ r'_{i,H} = b_i \\ r'_{i,L} = \overline{b_i} \cdot r_{i-1} \cdot r_{i-2} + \overline{b_i} \cdot r_i + b_i \cdot \overline{r_i} + b_i \cdot \overline{r_{i-1}} \cdot \overline{r_{i-2}} \end{cases}.$$

A hardware realization of these equations is given in Figure 8. Initially, the *shift-left registers* $\{r_i, r_{i-1}, r_{i-2}\}$ are loaded with $\{0, r_{m-1}, r_{m-2}\}$ and the *latch* is reset to logic "0". At the end of each iteration, the outputs $r'_{i,H}$ and $r'_{i,L}$ are used to encode the transformed SD2 digit r'_i. At this moment, the output b_{i-1} is fed into the latch, the lower bit r_{i-3} is prepared to be fed into the shift-left registers, and the registers shift one bit to the left.

Based on this minimum-weight signed-digit transformer, an hardware architecture for fast exponentiation, α^r, is sketched in Figure 9. In that exponentiation hardware, the outputs of the signed-digit transformer, $r'_{i,H}$ and $r'_{i,L}$, are used to control the computational data flow and the register fetching. Signal $r'_{i,H}$ is used to determine which of α or α^{-1} has to be selected by the *multiplexer* **MUX-1**, and signal $r'_{i,L}$ determines whether or not the multiplication $A \leftarrow A \times B$ (where A means the *accumulator* and B means the output of **MUX-1**) needs to be performed. In the case of $r'_{i,L} = 1$ (and thus $M' =$ "$0 \| 1$", where $\|$ means concatenation), both the squaring and the multiplication are required. On the other hand, when $r'_{i.L} = 0$ (and thus $M' = 0$), only the squaring operation is required.

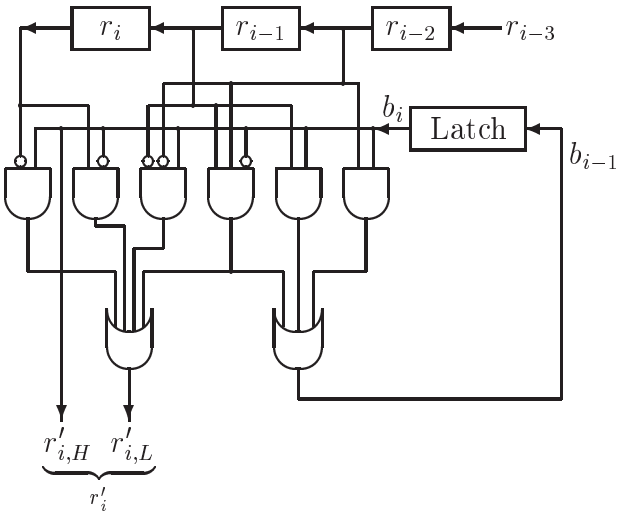


Fig. 8. Hardware implementation of the proposed transformer.

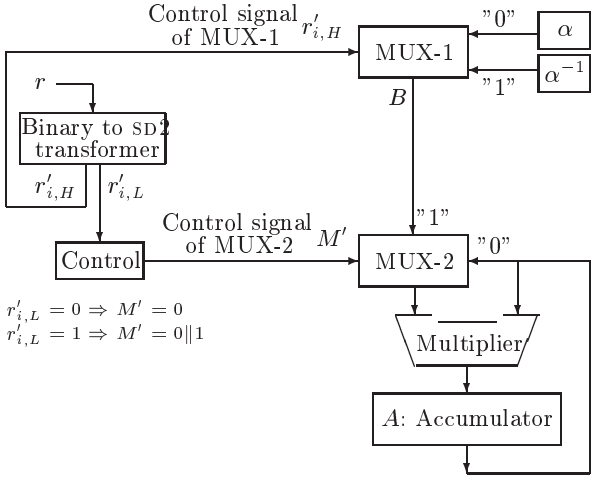


Fig. 9. Hardware architecture for the computation of α^r.

Using this architecture, the computation of α^r needs, on average, $\frac{4}{3}|r|$ multiplications (if we assume that multiplication and squaring are equally time-consuming operations). For example, α^r is evaluated after only 684 multiplications for a 512-bit integer r.

6 Other Considerations

For some special cases, the proposed algorithms may bring further advantages.

1. One of the main concerns of the proposed algorithms was to reduce the Hamming weight of exponent r for the computation of α^r. While the number of squarings still remains the same (i.e., $|r|$), the average number of multiplications was reduced from $|r|/2$ to $|r|/3$. Over GF(2^k), squaring operations can be achieved via a simple circular shift if the elements are represented in the so-called *normal basis* [28]. Their computational costs can therefore be ignored comparing to multiplications. In this case, the real expected gain over the binary methods is then $(\frac{1}{2} - \frac{1}{3})/\frac{1}{2} \approx 66.66\,\%$ instead of 11.11% as aforementioned in Section 2.
2. Because the expression of variable b, $b_{i-1} \leftarrow \lfloor (b_i + r_{i-1} + r_{i-2})/2 \rfloor$ (see Fig. 7), is similar to that of carry c in Reitwiesner's algorithm (Fig. 3), $c_{i+1} \leftarrow \lfloor (c_i + r_i + r_{i+1})/2 \rfloor$, the *parallel* recoding method proposed by Koç [29] readily extends to our minimum-weight left-to-right recoding algorithm.

7 Conclusions

To improve the performance of the square-and-multiply exponentiation for the evaluation of α^r, the Hamming weight of exponent r should be reduced. This can be achieved by adopting a signed-digit representation for exponent r. Assuming that α^{-1} is provided along with α (or can be cheaply evaluated, as for elliptic curves), a minimum-weight signed-digit representation for r can improve quite a large the computation of α^r. However, in order to produce the minimum-weight signed-digit representation, existing solutions must initiate the recoding process from *right to left* while only the modified *left-to-right* (MLR) exponentiation algorithm is suitable. The *heterogeneous* processings bring some time and memory inefficiencies for hardware realization. Indeed, the recoding has first to be performed, the resulting representation for r must be saved (which requires *twice* more memory space than its binary representation). Then and only then, the exponentiation process may start in order to obtain α^r.

In this paper, new and *homogeneous* approaches for *minimum-weight* signed-digit representation are presented. Our methods are homogeneous in the sense that both the proposed recoding algorithms and the MLR exponentation algorithm initiate from the most significant position of the exponent. The signed-digit representation of exponent r is consequently obtained in real-time and does not need to be stored. This better memory usage is especially useful for small devices like smart-cards. Our first algorithm gives the canonical representation for the

exponent. This algorithm is then modified into another minimum-weight recoding algorithm so that table look-up is possible. A hardware implementation of this second converter and the corresponding architecture for exponentiation are also presented.

Acknowledgments

We are grateful to Dan Gordon for sending a preprint of [16]. Many thanks also go to the anonymous reviewers for their excellent work.

This work was partly supported by the National Science Council of the Republic of China under contracts NSC89-2213-E-008-049, NSC87-2213-E-032-012, and NSC87-2811-E-032-0001.

References

1. A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electronic Computers*, vol. EC-10, pp. 389–400, 1961. Reprinted in [30, vol. II, pp. 54–65].
2. I. Koren, *Computer arithmetic algorithms*, Prentice-Hall, 1993.
3. N. Takagi, H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," *IEEE Trans. Computers*, vol. C-34, no. 9, pp. 789–796, 1985.
4. S. Kuninobu, T. Nishiyama, H. Edamatsu, T. Taniguchi, and N. Takagi, "Design of high speed MOS multiplier and divider using redundant binary representation," in *Proc. 8th Symp. Computer Arithmetic*, pp. 80–86, 1987.
5. Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," *IEEE J. Solid-State Circuits*, vol. 22, no. 1, pp. 28–34, 1987.
6. A. Vandemeulebroecke, E. Vanzieleghem, T. Denayer, and P.G.A. Jespers, "A new carry-free division algorithm and its application to a single-chip 1024-b RSA processor," *IEEE J. Solid-State Circuits*, vol. 25, no. 3, pp. 748–755, 1990.
7. K. Hwang, *Computer arithmetic, principles, architecture and design*, John Wiley & Sons, 1979.
8. S.M. Yen, C.S. Laih, C.H. Chen, and J.Y. Lee, "An efficient redundant-binary number to binary number converter," *IEEE J. Solid-State Circuits*, vol. 27, no. 1, pp. 109–112, 1992.
9. R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
10. Ç.K. Koç, "High-speed RSA implementations," Tech. Rep. TR 201, RSA Laboratories, Nov. 1994.
11. D.E. Knuth, *The art of computer programming/Seminumerical algorithms*, vol. 2, Addison-Wesley, 2nd edition, 1981.
12. P. Downey, B. Leong, and R. Sethi, "Computing sequences with addition chains," *SIAM J. Computing*, vol. 10, pp. 638–646, 1981.
13. J. Bos and M. Coster, "Addition chain heuristics," in *Advances in Cryptology — CRYPTO '89*, vol. 435 of *Lecture Notes in Computer Science*, pp. 400–407, Springer-Verlag, 1990.

14. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Information Theory*, vol. IT-31, no. 4, pp. 469–472, 1985.
15. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.
16. E.F. Brickell, D.M. Gordon, K.S. McCurley, and D.R. Wilson, "Fast exponentiation with precomputation: Algorithms and lower bounds," Preprint, Mar. 1995. An earlier version appeared in *Proc. of EUROCRYPT '92*.
17. F. Morain and J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains," *Theoretical Informatics and Applications*, vol. 24, pp. 531–543, 1990.
18. A.D. Booth, "A signed binary multiplication technique," *The Quaterly J. Mechanics and Applied Mathematics*, vol. 4, pp. 236–240, 1951. Reprinted in [30, vol. I, pp. 100–104].
19. G.W. Reitwiesner, "Binary arithmetic," *Advances in Computers*, vol. 1, pp. 231–308, 1960.
20. J. Jedwab and C. Mitchell, "Minimum weight modified signed-digit representations and fast exponentiation," *Electronics Letters*, vol. 25, pp. 1171–1172, 1989.
21. H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag, 1993.
22. Ö. Eğecioğlu and Ç.K. Koç, "Exponentiation using canonical recoding," *Theoretical Computer Science*, vol. 129, no. 2, pp. 407–417, 1994.
23. B.S. Kaliski, "The Montgomery inverse and its applications," *IEEE Trans. Computers*, vol. C-44, no. 8, pp. 1064–1065, 1995.
24. S. Arno and F.S. Wheeler, "Signed digit representations of minimal Hamming weight," *IEEE Trans. Computers*, vol. C-42, no. 8, pp. 1007–1010, 1993.
25. D.M. Gordon, "A survey of fast exponentiation methods," *J. Algorithms*, vol. 27, pp. 129–146, 1998.
26. W.E. Clark and J.J. Liang, "On arithmetic weight for a general radix representation of integers," *IEEE Trans. Information Theory*, vol. IT-19, pp. 823–826, 1973.
27. H. Wu and M.A. Hasan, "Efficient exponentiation of a primitive root in GF(2^m)," *IEEE Trans. Computers*, vol. C-46, no. 2, pp. 162–172, 1997.
28. J. Omura and J. Massey, "Computational method and apparatus for finite field arithmetic," U.S. Patent # 4,587,627, 1986.
29. Ç.K. Koç, "Parallel canonical recoding," *Electronics Letters*, vol. 32, pp. 2063–2065, 1996.
30. E.E. Swartzlander, Jr., Ed., *Computer arithmetic*, vol. I and II, IEEE Computer Society Press, 1990.

New Minimal Modified Radix-r Representation with Applications to Smart Cards

[Published in D. Naccache and P. Paillier, Eds., *Public Key Cryptography*, vol. 2274 of *Lecture Notes in Computer Science*, pp. 375–384, Springer-Verlag, 2002.]

Marc Joye[1] and Sung-Ming Yen[2]

[1] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos Cedex, France
E-mail: marc.joye@gemplus.com
http://www.gemplus.com/smart/ − http://www.geocities.com/MarcJoye/
[2] Laboratory of Cryptography and Information Security (LCIS)
Dept of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan 320, R.O.C.
E-mail: yensm@csie.ncu.edu.tw
http://www.csie.ncu.edu.tw/~yensm/

Abstract. This paper considers the problem of finding a minimum-weighted representation of an integer under any modified radix-r number system. Contrary to existing methods, the proposed transformation is carried out from the left to the right (i.e., from the most significant position). This feature finds numerous applications and especially in fast arithmetic techniques because it reduces both time and space complexities, which is particularly attractive for small devices like smart cards.

Keywords. Modified radix-r/signed-digit/minimum-weighted representations, generalized nonadjacent forms, integer recoding, smart cards, fast exponentiation, elliptic curves.

1 Introduction

A *modified radix-r representation* (MR-r) of an integer N is a sequence of digits $N = (\ldots, c_2, c_1, c_0)$ with $-r < c_i < r$. Unlike the (usual) radix-r representation (i.e., with $0 \leq c_i < r$), such a representation is not unique. The radix-r representation is a special case of MR-r representation.

In the theory of arithmetic codes [vL82] or for fast implementation of cryptosystems [Gor98, MvOV97], it is of interest to have a representation such that the number of nonzero digits (i.e., the *arithmetic weight* [CL73]) is minimal. In the binary case (i.e., when $r = 2$), a well-known minimal representation is given by the so-called *nonadjacent form* (NAF), that is, a representation with $c_i \cdot c_{i+1} = 0$ for all $i \geq 0$. In [Rei60], Reitwiesner proved that each integer has exactly one NAF. Clark and Liang [CL73] later addressed the general case and extended the notion of NAF to an arbitrary radix $r \geq 2$:

Definition 1. *A* MR-r *representation* $(\ldots, c_2, c_1, c_0)$ *is said to be a* generalized nonadjacent form *(*GNAF*) if and only if*

(G1) $|c_i + c_{i+1}| < r$ *for all* i *; and*
(G2) $|c_i| < |c_{i+1}|$ *if* $c_i \cdot c_{i+1} < 0$ *.*

As one can easily see, this form coincides with the definition of the NAF when $r = 2$. Moreover, as for the NAF, it can be proven that this form is unique and has minimal arithmetic weight [CL73].

However, the GNAF is not the only representation with minimal arithmetic weight. For example, $(1, 0, 1, 0, -1, 0)$ and $(1, 0, 0, 1, 1, 0)$ are two minimal MR-2 representations for 38.

Our results

This paper considers a new *minimal* MR-r representation and presents an efficient algorithm to compute it. This new representation, unlike the GNAF, presents the nice property to be obtained by scanning the digits *from the left to the right* (i.e., from the most significant digit to the least significant one). This processing direction is of great importance since only for that direction a table of precomputed values may be used to speed up the exponentiation, at least for exponentiation algorithms processing one digit at a time [BGMW92]. A subsequent advantage is that the exponent need not be recoded in advance, which results in better performances in both running time and memory space. This is especially important for small devices like the smart cards. Moreover, only for that direction, further speedups may be obtained if the element to be raised up presents a special structure [Coh93, pp. 9–10]. Finally, this processing direction also solves an open problem introduced in [WH97, § 3.6].

2 New Minimal Representation

Throughout this paper, for convenience, $-a$ will sometimes be denoted as $\bar{a}$ and $\langle S \rangle^k$ will represent $S, S, \ldots, S$ (k times).

2.1 Elementary blocks

Given the radix-r representation $(\ldots, c_2, c_1, c_0)$ of an integer N, the corresponding GNAF can easily be obtained as follows [CL73]: compute the radix-r representation of $(r+1)N$, $(\ldots, b_2, b_1, b_0)$, then the GNAF is given by $(\ldots, c'_2, c'_1, c'_0)$ where $c'_i = b_{i+1} - c_{i+1}$. So, if the radix-r representation of $(r+1)N$ is known, we are done. This computation can be carried out by right-to-left adding $N = (\ldots, c_2, c_1, c_0)$ and $rN = (\ldots, c_2, c_1, c_0, 0)$ according to the standard carry rule [Knu81, p. 251]:

$$\begin{cases} \kappa_0 = 0 \\ \kappa_{i+1} = \left\lfloor \dfrac{c_i + c_{i+1} + \kappa_i}{r} \right\rfloor \end{cases}, \tag{1}$$

$b_0 = c_0$ and $b_{i+1} = c_i + c_{i+1} + \kappa_i - r\kappa_{i+1}$ for $i \geq 0$. Left-to-right algorithms to add integers also exist [Knu81, Exercises 4.3.1.5 and 4.3.1.6]. However, they do not consistently output one digit per iteration; delay may occur when the sum of two adjacent digits is equal to $r-1$. For that reason, an *on-line* (i.e., digit-by-digit) left-to-right computation of the GNAF seems to be impossible.

This paper considers a quite different approach: instead of trying to obtain the GNAF, we are looking for other minimal forms that may be efficiently evaluated from the left to the right. Our technique relies on the following observation:

Proposition 1. *Let $(\ldots, c_2, c_1, c_0)$ be the radix-r representation of an integer N, and let $(c_{f+1}, c_f, \ldots, c_{e+1}, c_e)$ be a subchain of digits of this representation such that*

(E1) $f > e$;
(E2) $c_e + c_{e+1} \neq r-1$;
(E3) $c_j + c_{j+1} = r-1 \quad$ *for* $e < j < f$;
(E4) $c_f + c_{f+1} \neq r-1$.

Then all but the first and the last digits of the corresponding GNAF *are entirely determined.*

Proof. We note from Eq. (1) that, since $\kappa_i = 0$ or 1, the value of the carry-out κ_{i+1} does not depend on the carry-in κ_i for $i \geq e$. Indeed, we have $\kappa_{e+1} = \lfloor \frac{c_e + c_{e+1}}{r} \rfloor$ by Condition (E2); hence $\kappa_{j+1} = \kappa_{e+1}$ for $e \leq j < f$ by induction from Condition (E3); and $\kappa_{f+1} = \lfloor \frac{c_f + c_{f+1}}{r} \rfloor$ by Condition (E4). Therefore, if $(\star, c'_f, \ldots, c'_{e+1}, \star)$ denotes the digits corresponding to the GNAF, it follows that

$$\begin{aligned} c'_j &= b_{j+1} - c_{j+1} = c_j + \kappa_j - r\kappa_{j+1} \\ &= \begin{cases} c_j + (1-r)\kappa_{e+1} & \text{for } e+1 \leq j \leq f-1 \\ c_f + \kappa_{e+1} - r\kappa_{f+1} & \text{for } j = f \end{cases} \end{aligned} \tag{2}$$

where $\kappa_{e+1} = \lfloor \frac{c_e + c_{e+1}}{r} \rfloor$ and $\kappa_{f+1} = \lfloor \frac{c_f + c_{f+1}}{r} \rfloor$. □

A subchain of radix-r digits satisfying Conditions (E1)–(E4) will be called a *radix-r elementary block*. From Proposition 1, two types of elementary blocks may be distinguished.

Definition 2. *Let $0 \leq d, b, e \leq r-1$, $c = r-1-d$ (that is, $c + d = r-1$) and $k \geq 0$. An elementary block of the form*

$$(b, d, \langle c, d \rangle^k, e) \quad \textit{with } b + d \neq r-1 \textit{ and } e + d \neq r-1 \tag{3}$$

will be referred to as a Type 1 radix-r elementary block*; and an elementary block of the form*

$$(b, d, \langle c, d \rangle^k, c, e) \quad \textit{with } b + d \neq r-1 \textit{ and } e + c \neq r-1 \tag{4}$$

as a Type 2 radix-r elementary block*.*

Notice that a Type 1 elementary block contains an odd number of digits (and at least 3) while a Type 2 elementary block contains an even number of digits (and at least 4).

Based on this definition, we can now present the new recoding algorithm.

2.2 General recoding algorithm

From Proposition 1 (see also Eq. (2)), the GNAF corresponding to a Type 1 elementary block $(b, d, \langle c, d\rangle^k, e)$ is given by[1]

$$G_1 = (\star, d + \lfloor \tfrac{e+d}{r} \rfloor - r\lfloor \tfrac{b+d}{r} \rfloor, \langle c + (1-r)\lfloor \tfrac{e+d}{r} \rfloor, d + (1-r)\lfloor \tfrac{e+d}{r} \rfloor \rangle^k, \star) \ . \quad (5)$$

By definition of a Type 1 elementary block, we have $b + d \neq r - 1$ and $e + d \neq r - 1$. Hence, Eq. (5) respectively simplifies to

$$G_1 = \begin{cases} (\star, \langle d, c, \rangle^k, d, \star) & \text{if } b + d < r - 1 \text{ and } e + d < r - 1 \\ (\star, d + 1, \langle -d, -c\rangle^k, \star) & \text{if } b + d < r - 1 \text{ and } e + d \geq r \\ (\star, d - r, \langle c, d\rangle^k, \star) & \text{if } b + d \geq r \text{ and } e + d < r - 1 \\ (\star, -c, \langle -d, -c\rangle^k, \star) & \text{if } b + d \geq r \text{ and } e + d \geq r \end{cases} . \quad (6)$$

Similarly, the GNAF corresponding to a Type 2 block $(b, d, \langle c, d\rangle^k, c, e)$ is given by

$$G_2 = \begin{cases} (\star, d, \langle c, d\rangle^k, c, \star) & \text{if } b + d < r - 1 \text{ and } e + c < r - 1 \\ (\star, d + 1, \langle -d, -c\rangle^k, -d, \star) & \text{if } b + d < r - 1 \text{ and } e + c \geq r \\ (\star, d - r, \langle c, d\rangle^k, c, \star) & \text{if } b + d \geq r \text{ and } e + c < r - 1 \\ (\star, -c, \langle -d, -c\rangle^k, -d, \star) & \text{if } b + d \geq r \text{ and } e + c \geq r \end{cases} . \quad (7)$$

Here too, we see the difficulty of converting a radix-r representation into its GNAF by scanning the digits from the left to the right. If an elementary block begins with $(b, d, c, d, c, d, \ldots)$ with $b + d < r - 1$, the output will be $(\star, d, c, d, c, d, \ldots)$ or $(\star, d+1, -d, -c, -d, -c, \ldots)$ depending on the two last digits of the elementary block; if $b + d \geq r$, the output will be $(\star, d - r, c, d, c, d, \ldots)$ or $(\star, -c, -d, -c, -d, -c, \ldots)$. However, as stated in the next lemma, these outputs may be replaced by *equivalent* forms (i.e., forms having the same length and the same weight, and representing the same integer) so that the two last digits of an elementary block do not need to be known in advance.

Lemma 1. *Let $0 \leq d \leq r - 1$ and $c = r - 1 - d$. Then*

(i) if $d \neq r - 1$, $(d + 1, \langle -d, -c\rangle^k)$ and $(\langle d, c\rangle^k, d + 1)$ are equivalent MR-r *representations;*

(ii) if $d \neq 0$, $(d - r, \langle c, d\rangle^k)$ and $(\langle -c, -d\rangle^k, d - r)$ are equivalent MR-r *representations.*

Proof. With the MR-r notation, $(d + 1, \langle -d, -c\rangle^k)$ represents the integer

$$\begin{aligned} N &= (d+1)r^{2k} + \textstyle\sum_{j=0}^{k-1}(-dr - c)r^{2j} \\ &= (d+1)r^{2k} + \textstyle\sum_{j=0}^{k-1}(cr + d + 1 - r^2)r^{2j} \\ &= d\,r^{2k} + \textstyle\sum_{j=0}^{k-1}(cr + d)r^{2j} + 1 \quad \text{since } \textstyle\sum_{j=0}^{k-1} r^{2j} = \frac{r^{2k}-1}{r^2-1} \end{aligned}$$

which is thus also represented by $(d, \langle c, d\rangle^{k-1}, c, d + 1)$. We also note that $(d + 1, \langle -d, -c\rangle^k)$ and $(\langle d, c\rangle^k, d+1)$ have the same arithmetic weight since their digits are identical, in absolute value. The second equivalence is proved similarly. □

[1] The '$\star$' indicates that the digit is unknown.

Consequently, GNAFs G_1 and G_2 (see Eqs (6) and (7)) can respectively be replaced by another equivalent form, which we call *generalized star form* (GSF), respectively given by

$$S_1 = \begin{cases} (\star, \langle d, c\rangle^k, d, \star) & \text{if } b+d < r-1 \text{ and } e+d < r-1 \\ (\star, \langle d, c\rangle^k, d+1, \star) & \text{if } b+d < r-1 \text{ and } e+d \geq r \\ (\star, \langle -c, -d\rangle^k, d-r, \star) & \text{if } b+d \geq r \text{ and } e+d < r-1 \\ (\star, \langle -c, -d\rangle^k, -c, \star) & \text{if } b+d \geq r \text{ and } e+d \geq r \end{cases} \tag{8}$$

and

$$S_2 = \begin{cases} (\star, \langle d, c\rangle^k, d, c, \star) & \text{if } b+d < r-1 \text{ and } e+c < r-1 \\ (\star, \langle d, c\rangle^k, d+1, -d, \star) & \text{if } b+d < r-1 \text{ and } e+c \geq r \\ (\star, \langle -c, -d\rangle^k, d-r, c, \star) & \text{if } b+d \geq r \text{ and } e+c < r-1 \\ (\star, \langle -c, -d\rangle^k, -c, -d, \star) & \text{if } b+d \geq r \text{ and } e+c \geq r \end{cases}. \tag{9}$$

The outputs now behave very regularly; an elementary block $(b, d, c, d, c, d, \ldots)$ will be recoded into $(\star, d, c, d, c, d, \ldots)$ if $b+d < r-1$ or into $(\star, -c, -d, -c, -d, -c, \ldots)$ if $b+d \geq r$. We have just to take some precautions when outputting the last digits of the corresponding GSFs. The following example clarifies the technique.

Example 1. Suppose that the radix-4 GSF of $N = 208063846$ has to be computed. Its radix-4 representation is $(30121230311212)_4$. Then, by adding artificial beginning and ending 0's and decomposing this representation into elementary blocks, the corresponding GSF is easily obtained from Eqs (8) and (9) as follows.

Radix-4 representation:	0 0 3 0 1 2 1 2 3 0 3 1 1 2 1 2 .0
Elementary blocks:	0 0 3 0 1
	0 1 2 1 2 3
	2 3 0 3 1
	3 1 1
	1 1 2 1 2 .0
Corresponding GSF blocks:	$\star$ 0 3 0 $\star$
	$\star$ 1 2 2 $\bar{1}$ $\star$
	$\star$ 0 $\bar{3}$ 0 $\star$
	$\star$ $\bar{3}$ $\star$
	$\star$ 1 2 1 2 $\star$
Radix-4 GSF representation:	3 0 1 2 2 $\bar{1}$ 0 $\bar{3}$ 0 $\bar{3}$ 1 2 1 2

□

In the previous example, it clearly appears that, from the definition of an elementary block, the two first digits of an elementary block are the two last ones of the previous block. This also illustrates that the decomposition into elementary (and thus GSF) blocks always exists. Note also that the values of the first and last digits of the corresponding GSF are given by the adjacent GSF blocks.

So, by carefully considering the two last digits of each possible elementary block, we finally obtain the desired algorithm. Two additional variables are used: variable β_i plays a role similar to the "borrow" and τ keeps track the value of the repeated digit (d in Definition 2).

```
INPUT:  (n_{m-1}, ..., n_0)      (Radix-r representation)
OUTPUT: (n'_m, n'_{m-1}, ..., n'_0) (Radix-r GSF representation)
  β_m ← 0; n_m ← 0; n_{-1} ← 0; n_{-2} ← 0, τ ← 0
  for i from m down to 0 do
    case
      n_i + n_{i-1} < r - 1: β_{i-1} ← 0, τ ← n_{i-1}
      n_i + n_{i-1} = (r - 1): β_{i-1} ← { 0    if β_i = 1, n_{i-1} = (r - 1 - τ)
                                         {      and n_{i-1} + n_{i-2} < r - 1
                                         { 1    if β_i = 0, n_{i-1} = (r - 1 - τ)
                                         {      and n_{i-1} + n_{i-2} ≥ r
                                         { β_i  otherwise
      n_i + n_{i-1} ≥ r: β_{i-1} ← 1, τ ← n_{i-1}
    endcase
    n'_i ← -rβ_i + n_i + β_{i-1}
  od
```

Fig. 1. Left-to-right radix-r GSF recoding algorithm.

To verify the correctness of the algorithm, it suffices to check that each type of elementary block is effectively transformed accordingly to Eqs (8) and (9). The next theorem states that the proposed algorithm is optimal in the sense that the resulting representation has the fewest number of nonzero digits of any MR-r representation.

Theorem 1. *The* GSF *of a number has minimal arithmetic weight.*

Proof. This is straightforward by noting the GSF is obtained from the GNAF where some subchains were replaced according to Lemma 1. Since these transformations produce equivalent subchains, the GSF has the same arithmetic weight as the GNAF and is thus minimal. □

Remark 1. We note that, as in [CL73], the proposed algorithm can be extended to transform an arbitrary MR-r representation into a minimal one.

2.3 Binary case

In the binary case (i.e., when $r = 2$), the algorithm presented in Fig. 1 is slightly simpler. Using the same notations as in Fig. 1, if $n_i + n_{i-1} = 1$, we can easily verify that

$$\beta_{i-1} = \left\lfloor \frac{\beta_i + n_{i-1} + n_{i-2}}{2} \right\rfloor \tag{10}$$

is a valid expression for β_{i-1}. Moreover, this expression remains valid when $n_i = n_{i-1} = 0$ (i.e., in the case $n_i + n_{i-1} < 1$) because if $n_{i+1} = 0$ then $\beta_i = 0$ and if $n_{i+1} = 1$ then, by Eq. (10), we also have $\beta_i = 0$; therefore Eq. (10) yields $\beta_{i-1} = 0$. Similarly when $n_i = n_{i-1} = 1$ (i.e., $n_i + n_{i-1} \geq 2$), we can show that Eq. (10) yields $\beta_{i-1} = 1$, as expected. The radix-2 GSF recoding algorithm thus becomes [JY00]:

```
INPUT:  (n_{m-1}, ..., n_0)      (Binary representation)
OUTPUT: (n'_m, n'_{m-1}, ..., n'_0)  (Binary GSF representation)
  β_m ← 0; n_m ← 0; n_{-1} ← 0; n_{-2} ← 0
  for i from m down to 0 do
    β_{i-1} ← ⌊(β_i + n_{i-1} + n_{i-2})/2⌋
    n'_i ← -rβ_i + n_i + β_{i-1}
  od
```

Fig. 2. Left-to-right binary GSF recoding algorithm.

3 Applications

Minimal representations naturally find applications in the theory of arithmetic codes [vL82] and in fast arithmetic techniques [Boo51, Avi61]. In this section, we will restrict our attention to fast exponentiation (see the excellent survey article of [Gor98]).

The commonly used methods to compute g^k are the generalized "square-and-multiply" algorithms. When base g is fixed or when the computation of an inverse is virtually free, as for elliptic curves [MO90], a signed-digit representation further speeds up the computation. These algorithms input the MR-r representations of exponent $k = (k_m, \ldots, k_0)$ (i.e., with $-r < k_i < r$ and $k_m \neq 0$) and of base g, and output $y = g^k$. This can be carried out from right to left (Fig. 3a) or from left to right (Fig. 3b).

```
INPUT:  k = (k_m, ..., k_0), g
OUTPUT: y = g^k
  y ← 1; h ← g
  for i from 0 to m - 1 do
    y ← y · h^{k_i}
    h ← h^r
  od
  y ← y · h^{k_m}
```

(a) Right-to-left (RL).

```
INPUT:  k = (k_m, ..., k_0), g
OUTPUT: y = g^k
  y ← g^{k_m}
  for i from m - 1 down to 0 do
    y ← y^r
    y ← y · g^{k_i}
  od
```

(b) Left-to-right (LR).

Fig. 3. Modified exponentiation algorithms.

Using Markov chains, Arno and Wheeler [AW93] precisely estimated that the average proportion of nonzero digits in the radix-r GNAF (and thus also in the radix-r GSF) representation is equal to

$$\rho_r = \frac{r-1}{r+1} \ . \tag{11}$$

The LR algorithm can use the precomputation of some g^t for $-r < t < r$ while the RL algorithm cannot! One could argue that the exponent may first be recoded into a minimal MR-r representation and then the LR algorithm be applied. However this also requires more memory space since each digit in the MR-r requires an additional bit (to encode the sign for each digit). Consequently, since only the LR algorithm is efficient, the proposed (left-to-right) general recoding algorithm (Fig. 1) will also bring some advantages because, as aforementioned, the exponent need not be pre-recoded into its MR-r representation. This feature is especially desirable for small devices, like smart cards.

Another generalization of exponentiation algorithms is to use variable windows to skip strings of consecutive zeros. Here too, the proposed recoding algorithms offer some speedups. Indeed, as a side effect, while having the same weight as the GNAF, the GSF representation has more adjacent nonzero digits, which increases the length of the runs of zeros. This property was successfully applied by Koyama and Tsuruoka [KT93] to speed up the window exponentiation on elliptic curves. We note nevertheless that the zero runs of their algorithm are generally longer than those obtained by the GSFs.

4 Conclusions

In this paper, a new modified representation of integers for a general radix $r \geq 2$ is developed. Like the nonadjacent form (NAF) and its generalization, the proposed representation has the fewest number of nonzero digits of any modified representation.

When developing fast computational algorithms for hardware implementation, especially for some small devices which have very limited amount of memory, the problem of extra space requirement cannot be overlooked. Scanning and transforming the original representation from the most significant digit has its merit in not storing the resulting minimum-weighted result, hence memory space requirements are reduced. Finally, this solves a problem considered to be hard (see [WH97, § 3.6]), i.e., to obtain a minimal modified radix-4 representation by scanning the digits of the standard representation from the left to the right.

Acknowledgments. The authors are grateful to the anonymous referees for their useful comments. Sung-Ming Yen was supported in part by the National Science Council of the Republic of China under contract NSC 87-2213-E-032-013.

References

[Avi61] A. Avizienis, *Signed-digit number representations for fast parallel arithmetic*, IRE Transactions on Electronic Computers **EC-10** (1961), 389–400, Reprinted in [Swa90, vol. II, pp. 54–65].

[AW93] S. Arno and F.S. Wheeler, *Signed digit representations of minimal Hamming weight*, IEEE Transactions on Computers **C-42** (1993), no. 8, 1007–1010.

[BGMW92] E.F. Brickell, D.M. Gordon, K.S. McCurley, and D.B. Wilson, *Fast exponentiation with precomputation*, Advances in Cryptology – EUROCRYPT '92, Lecture Notes in Computer Science, vol. 658, Springer-Verlag, 1992, pp. 200–207.

[Boo51] A.D. Booth, *A signed binary multiplication technique*, The Quaterly Journal of Mechanics and Applied Mathematics **4** (1951), 236–240, Reprinted in [Swa90, vol. I, pp. 100–104].

[CL73] W.E. Clark and J.J. Liang, *On arithmetic weight for a general radix representation of integers*, IEEE Transactions on Information Theory **IT-19** (1973), 823–826.

[Coh93] H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag, 1993.

[EK94] Ö. Eğecioğlu and Ç.K. Koç, *Exponentiation using canonical recoding*, Theoretical Computer Science **129** (1994), no. 2, 407–417.

[GHM96] D. Gollmann, Y. Han, and C.J. Mitchell, *Redundant integer representation and fast exponentiation*, Designs, Codes and Cryptography **7** (1996), 135–151.

[Gor98] D.M. Gordon, *A survey of fast exponentiation methods*, Journal of Algorithms **27** (1998), 129–146.

[JY00] M. Joye and S.-M. Yen, *Optimal left-to-right binary signed-digit recoding*, IEEE Transactions on Computers **49** (2000), no. 7, 740–748.

[Knu81] D.E. Knuth, *The art of computer programming/Seminumerical algorithms*, 2nd ed., vol. 2, Addison-Wesley, 1981.

[KT93] K. Koyama and Y. Tsurukoa, *Speeding up elliptic cryptosystems by using a signed binary window method*, Advances in Cryptology – CRYPTO '92, Lecture Notes in Computer Science, vol. 740, Springer-Verlag, 1993, pp. 345–357.

[MO90] F. Morain and J. Olivos, *Speeding up the computations on an elliptic curve using addition-subtraction chains*, Theoretical Informatics and Applications **24** (1990), 531–543.

[MvOV97] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of applied cryptography, ch. 14, CRC Press, 1997.

[Rei60] G.W. Reitwiesner, *Binary arithmetic*, Advances in Computers **1** (1960), 231–308.

[Swa90] E.E. Swartzlander, Jr. (ed.), *Computer arithmetic*, vol. I and II, IEEE Computer Society Press, 1990.

[vL82] J.H. van Lint, *Introduction to coding theory*, Springer-Verlag, 1982.

[WH97] H. Wu and M.A. Hasan, *Efficient exponentiation of a primitive root in* $\mathrm{GF}(2^m)$, IEEE Transactions on Computers **C-46** (1997), no. 2, 162–172.

Efficient Generation of Prime Numbers*

[Published in Ç.K. Koç and C. Paar, Eds., *Cryptographic Hardware and Embedded Systems – CHES 2000*, vol. 1965 of *Lecture Notes in Computer Science*, pp. 340–354, Springer-Verlag, 2000.]

Marc Joye[1], Pascal Paillier[1], and Serge Vaudenay[2]

[1] Gemplus Card International, France
{marc.joye, pascal.paillier}@gemplus.com
[2] École Polytechnique Fédérale de Lausanne, Switzerland
Serge.Vaudenay@epfl.ch

Abstract. The generation of prime numbers underlies the use of most public-key schemes, essentially as a major primitive needed for the creation of key pairs or as a computation stage appearing during various cryptographic setups. Surprisingly, despite decades of intense mathematical studies on primality testing and an observed progressive intensification of cryptographic usages, prime number generation algorithms remain scarcely investigated and most real-life implementations are of rather poor performance. Common generators typically output a n-bit prime in heuristic average complexity $O(n^4)$ or $O(n^4/\log n)$ and these figures, according to experience, seem impossible to improve significantly: this paper rather shows a simple way to substantially reduce the value of hidden constants to provide much more efficient prime generation algorithms. We apply our techniques to various contexts (DSA primes, safe primes, ANSI X9.31-compliant primes, strong primes, etc.) and show how to build fast implementations on appropriately equipped smart-cards, thus allowing on-board key generation.

Keywords. Prime number generation, key generation, RSA, DSA, fast implementations, crypto-processors, smart-cards.

1 Introduction

Traditional prime number generation algorithms asymptotically require $O(n^4)$ or $O(n^4/\log n)$ bit operations where n is the bit-length of the expected prime number. This complexity may even become of the order of $O(n^5/(\log n)^2)$ in the case of *constrained* primes, such as safe or quasi-safe primes for instance. These asymptotic behaviors,[1] according to experience, seem impossible to improve significantly. In this paper, we rather propose simple algebraic methods

* Some parts presented in this paper are patent pending.

[1] assuming that multiplications modulo q are in $O(|q|^2)$. Theoretically, one could decrease this complexity by using multiplication algorithms such as Karatsuba in $O((|q|^{\log_2 3})$ or Schönhage-Strassen in $O(|q| \log |q| \log\log |q|)$.

which substantially reduce the value of the hidden constants, thus providing much more efficient prime generation algorithms.

We apply our techniques to various contexts such as DSA primes [9], strong primes [14] and ANSI X9.31-compliant primes [1], that is, real-life scenarios of well-recognized utility. As an illustration, we also reduce the number of rounds of Boneh and Franklin's [3] shared RSA keys protocol by a factor of nearly 10.

Finally, our techniques allow fast implementations on cryptographic smart-cards for on-board RSA [15] (or other schemes) key generation. Our motivation here is to help transferring this task from terminals to smart-cards themselves in the near future for more confidence, security, and compliance with network-scaled distributed protocols that include smart-cards, such as electronic cash or mobile commerce.

Notations. Throughout this paper, the following notations are used. Other notations will be introduced when needed.

Symbol	Signification
$\#\mathcal{A}$	cardinal of a set $\mathcal{A}$
$\|x\|$	bit-length of number x
$\{0,1\}^t$	set of t-bit numbers
$\mathbb{Z}_{\Pi}$	ring of integers modulo Π
$\mathbb{Z}_{\Pi}^*$	multiplicative group of $\mathbb{Z}_{\Pi}$
$\phi(\cdot)$	Euler's totient function
$\lambda(\cdot)$	Carmichael's function
$\mathrm{O}(\cdot)$	asymptotic bound
$\Omega(\cdot)$	asymptotic equivalence
$a \approx b$	a is approximatively equal to b
$a \gtrsim b$	$a \approx b$ and $a \geq b$
$a \lesssim b$	$a \approx b$ and $a \leq b$

2 Primality and Compositeness Tests

A lot of studies on primality testing have been carried out for years, and can be found in the literature devoted to the subject (e.g., see [7]). Computationally, we may distinguish *true primes* and *probable primes*: the difference being the way these are generated. A probable prime is usually obtained through a *compositeness test*. Such a test declares that a number is composite with probability 1 or prime with some probability < 1. Hence repeatedly running the test gives more and more confidence in the generated (probable) prime. Typical examples of compositeness tests include Fermat test, Solovay-Strassen test [16], and Miller-Rabin test [10, p. 379].

There also exist (true) *primality tests*, which declare a number prime with probability 1 (e.g., Pocklington's test [12] and its elliptic curve analogue [2], the Jacobi sum test [4]). However, these tests are generally more expensive or intricate.

To motivate further analysis, we hereafter assume that we are given some compositeness test T provided as a primality oracle of complexity $\tau(n) = \mathrm{O}(n^3)$ and of negligible error probability. Designing an efficient prime generation algorithm then reduces to the problem of knowing how to use T in order to produce a n-bit prime with a minimal number of calls to the oracle.

3 Generating Primes: Prior Art

3.1 Naive generators

We refer to the naive prime number generator as the following:

1. pick a random n-bit odd number q
2. if $\mathsf{T}(q) =$ `false` then goto 1
3. output q

Fig. 1. Naive prime number generator.

Neglecting calls to the random number generator, the expected number of trials here is asymptotically equal to $(\ln 2^n)/2 \approx 0.347\, n$. Generating a 256-bit prime thus requires 89 trials in average.

The previous algorithm has an incremental variant, which is given below on Fig. 2.

1. pick a random n-bit odd number q
2. while $\mathsf{T}(q) =$ `false` do $q \leftarrow q + 2$
3. output q

Fig. 2. Naive incremental prime number generator.

It should be outlined that this second algorithm has not the same proven complexity [5]. A proper analysis actually has to exploit the properties of the distribution of prime numbers, in connection with Riemann's Hypothesis. The incremental generator is however commonly used and we recall that it was shown to fail with probability $\mathrm{O}(t^3\, 2^{-\sqrt{t}})$ after $\Omega(t)$ trials (see [11, p. 148]).

3.2 Classical generation algorithms

The naive incremental generator can be made more efficient by choosing the initial candidate q already co-prime to small primes. Usually, one defines $\Pi =$

$2 \cdot 3 \cdots 29$ and randomly chooses a n-bit number q satisfying $\gcd(q, \Pi) = 1$. If $\mathsf{T}(q) = \texttt{false}$ then q is updated as $q \leftarrow q + \Pi$ (note that the naive generator corresponds to the special case $\Pi = 2$). If Π is a constant independent from n and contains k distinct primes, we denote this probabilistic algorithm by $\mathsf{H}[n, k]$.

The next section presents two new algorithms. The first one, making use of look-up tables, produces random numbers *constructively* co-prime to small primes. The second algorithm, slightly slower, is space-optimized and particularly suited for smart-card implementations. Based on this, we construct a new prime generation algorithm in Section 5 and give timing results in Section 6. Finally, in Section 7, we apply these new techniques to particular contexts.

4 Generating Invertible Numbers modulo a Product of Primes

Common prime number generators generally include a stage of trial divisions by small primes. We investigate in this section a way of avoiding this stage by efficiently constructing candidates that already satisfy co-primality properties. We base our constructions on simple algebraic techniques.

4.1 A table-based method

Let $\Pi = \prod_{i=1}^{k} {p_i}^{\delta_i}$ be a n-bit product of the first k primes with some small exponents δ_i. Let $\Delta = \max_i \delta_i$. We denote by $x = (x_1, \ldots, x_k)_{\equiv}$ the modular representation of $x \in \mathbb{Z}_\Pi$, i.e., $x_i = x \bmod p_i^{\delta_i}$. For $i = 1, \ldots, k$, one then defines $\theta_i = (0, \ldots, 1, \ldots, 0)_{\equiv}$ where the "1" stands in i^{th} position. It is obvious to see that we always have

$$\forall x \in \mathbb{Z}_\Pi \qquad x = \sum_{i=1}^{k} x_i \, \theta_i \bmod \Pi \ ,$$

that is, the function $x \mapsto (x_i)$ is a bijection[2] from $\mathbb{Z}_{p_1^{\delta_1}} \times \cdots \times \mathbb{Z}_{p_k^{\delta_k}}$ into $\mathbb{Z}_\Pi$. This function also defines a bijection from $\mathbb{Z}^*_{p_1^{\delta_1}} \times \cdots \times \mathbb{Z}^*_{p_k^{\delta_k}}$ to $\mathbb{Z}^*_\Pi$, and it follows that

$$\begin{aligned} \forall x \in \mathbb{Z}_\Pi \qquad x \in \mathbb{Z}^*_\Pi &\iff x_i \in \mathbb{Z}^*_{p_i^{\delta_i}} \\ &\iff x_i^{\delta_i} \not\equiv 0 \pmod{p_i^{\delta_i}} \\ &\iff x_i^{\delta_i} \theta_i \not\equiv 0 \pmod{\Pi} \quad \text{for } i = 1, \ldots, k \ . \end{aligned} \tag{1}$$

As a consequence, it appears that $x \in \mathbb{Z}^*_\Pi$ can be built-up from numbers x_i as long as they verify Eq. (1) above. We then define $\mathcal{A}$ as a set of random sequences $\alpha = (\alpha_1, \alpha_2, \ldots)$ with $\alpha_i \in \{0,1\}^t$. Equation (1) gives a natural way of surjectively transforming any $\alpha \in \mathcal{A}$ into an invertible number $\mathsf{g}(\alpha) \in \mathbb{Z}^*_\Pi$. The corresponding algorithm, g, is depicted on Fig. 3.

[2] This is the usual Chinese Remainder Theorem correspondence [8].

Precomputations: $\Pi = \prod_{i=1}^{k} p_i^{\delta_i}$, $t = C \cdot \max_i |p_i^{\delta_i}|$ $(C = 2)$, $\{\theta_i\}$
Input: a random sequence α
Output: an invertible number c modulo Π

1. $c = 0$
2. for $i = 1$ to k
   - 2.1 pick a random t-bit number α_i from α
   - 2.2 if $\alpha_i^{\delta_i}\theta_i \bmod \Pi = 0$ goto 2.1
   - 2.3 $c \leftarrow c + \alpha_i\,\theta_i \bmod \Pi$
3. output c

Fig. 3. Generator g of invertible numbers modulo Π.

Since we use t-bit numbers α_i and reduce them (implicitly) modulo $p_i^{\delta_i}$, there exists a bias (lying around 2^{-t}) leading to a non-uniform output distribution.[3] But this underlying bias may easily be made arbitrarily small by increasing t (which negatively affects the average complexity as well). We therefore suggest $t = C \cdot \max_i |p_i^{\delta_i}|$ as a good compromise, where the ratio C may be fixed to 2 for practical implementations. Further, we claim (for a negligible bias) that the function $\mathsf{g} : \mathcal{A} \to \mathbb{Z}_\Pi^*$ verifies

(*i*) to be surjective;
(*ii*) that for each element x of $\mathbb{Z}_\Pi^*$, the number of x's pre-images is about $\#\mathcal{A}/\#\mathbb{Z}_\Pi^* = \#\mathcal{A}/\,\phi(\Pi)$, and this guarantees the uniformity of g's outputs from its inputs;
(*iii*) g has a low (time) complexity $\gamma(n) = \mathrm{O}(n^2)$.

4.2 Modular search method

As aforementioned, the algorithm g generates uniformly distributed elements of $\mathbb{Z}_\Pi^*$. Although the execution time of g happens to be excellent when using an arithmetic processor, the memory space needed to store the numbers $\{\theta_i\}$ may appear dissuasive, in particular on a smart-card where memory may be subject to strong size constraints. We propose here a simple alternative method based on Carmichael's theorem[4]

$$\forall c \in \mathbb{Z}_\Pi^* \qquad c^{\lambda(\Pi)} \equiv 1 \pmod{\Pi}\,,$$

or more exactly on its converse:

Proposition 1. *$\forall c \in \mathbb{Z}_\Pi$, if $c^{\lambda(\Pi)} \equiv 1 \pmod{\Pi}$ then $c \in \mathbb{Z}_\Pi^*$.*

[3] It nevertheless seems a hard task to exploit it in some way for *a posteriori* secret key retrieval.

[4] If $\Pi = \prod_{i=1}^{k} p_i^{\delta_i}$ then $\lambda(\Pi) = \mathrm{lcm}[\lambda(p_i^{\delta_i})]_{i=1,\ldots,k}$, and $\lambda(p_i^{\delta_i}) = \phi(p_i^{\delta_i}) = p_i^{\delta_i - 1}(p_i - 1)$ for an odd prime p_i, $\lambda(2) = 1$, $\lambda(4) = 2$ and $\lambda(2^{\delta_i}) = \frac{1}{2}\,\phi(2^{\delta_i}) = 2^{\delta_i - 2}$ for $\delta_i \geq 3$.

Proof. A number $0 \leq c < \Pi$ is in $\mathbb{Z}_{\Pi}^{*}$ if and only if, for all primes p dividing Π, we have $\gcd(c, p) = 1 \iff c^{p-1} \equiv 1 \pmod{p}$ so that $c^{\lambda(\Pi)} \equiv 1 \pmod{\Pi}$ by Chinese remaindering. □

This provides an easy co-primality test that requires a single modular exponentiation with exponent $\lambda(\Pi)$. Note that this technique only needs the storage of Π and $\lambda(\Pi)$, and is also particularly suitable for crypto-processors. In addition, since Π is smooth, $\lambda(\Pi)$ is optimally small. The obtained procedure is depicted below.

Precomputations: $\Pi = \prod_{i=1}^{k} p_i^{\delta_i}$ and $\lambda(\Pi)$
Output: an invertible number c modulo Π
1. pick an n-bit random number $c < \Pi$
2. while $c^{\lambda(\Pi)} \bmod \Pi \neq 1$ do $c \leftarrow c + 1$
3. output c

Fig. 4. Generator g' of invertible numbers modulo Π.

The previous algorithm can be improved via Chinese remaindering. Instead of testing the co-primality of c to Π, one checks the co-primality to some factor of Π, say π_1. If $\gcd(c, \pi_1) \neq 1$ (i.e., if $c^{\lambda(\pi_1)} \bmod \pi_1 \neq 1$) then we already know that $\gcd(c, \Pi) \neq 1$. Otherwise, we test the co-primality of c with another factor π_2 of Π, and so on for several factors π_i until $\prod_i \pi_i = \Pi$ (or is a multiple of Π).

Although the complexity of g' may appear greater than $\gamma(n)$, the comparison must take into account the computational features of the underlying crypto-processor (see Section 6). Of course, the implementer shall choose between generators g and g' (or a variant) according to the necessity of saving time (using g) or space (using g'). We consider in the following that this choice has been done once for all, and that a black-box generator (hereafter referred to as g) of elements of $\mathbb{Z}_{\Pi}^{*}$ is at disposal: we now have to deal with how to design a prime generation algorithm in which primitives T and g get optimally exploited.

5 An Efficient Prime Generation Algorithm

Generated primes are expected to lie in some target window $\mathcal{F} = [w_{\min}, w_{\max}]$, where $w_{\max} = 2^n - 1$ in most contexts, and $w_{\min}$ is equal to $2^{n-1} + 1$ when generating n-bit primes, or $\left\lceil \sqrt{2^{2n-1} + 1} \right\rceil$ if the context imposes to obtain a strict $2n$-bit number when multiplying two so-generated primes (RSA moduli for instance).

The basic idea consists in utilizing g to produce a sequence of candidates that will be tested one by one until a prime is found. We now describe how we choose parameter Π. First, we find an integer η containing a maximum number

Output: a n-bit prime q
1. $c = \mathsf{g}()$
2. $q = c + \rho$
3. if $\mathsf{T}(q) = \texttt{false}$ then $c \leftarrow f_a(c)$ and goto 2.
4. output q

Fig. 5. $\mathsf{G}[n]$ – A basic prime number generator based on g.

of (different) primes (or more precisely minimizing the ratio $\phi(\eta)/\eta$) and such that there exist small integers $\varepsilon_{\min}$ and $\varepsilon_{\max}$ satisfying

$$\varepsilon_{\min}\eta \gtrapprox w_{\min} \quad \text{and} \quad \varepsilon_{\max}\eta \lessapprox w_{\max} - w_{\min} \ .$$

We then set

$$\Pi = \varepsilon_{\max}\,\eta \quad \text{and} \quad \rho = \varepsilon_{\min}\,\eta \ . \tag{2}$$

Once an invertible element $c^{(1)} \in \mathbb{Z}_\Pi^*$ is generated (using g), the first prime candidate is defined as

$$q^{(1)} = c^{(1)} + \rho \ .$$

Note that $\gcd(q^{(1)}, \eta) = \gcd(c^{(1)} + \rho, \eta) = \gcd(c^{(1)}, \eta) = 1$ since $c^{(1)} \in \mathbb{Z}_\Pi^*$; note also that $q^{(1)} \in \mathcal{F}$. We let $\mathcal{P}_0$ denote the set $(\mathbb{Z}_\Pi^* + \rho) \subseteq \mathcal{F}$, and $\mathcal{P}_c$ the set of primes belonging to $\mathcal{P}_0$. For avoiding systematic use of g, rejected candidates should optimally be transformed and re-used in order to continue the search. In this setting, the transition step $c^{(i+1)} = f_a(c^{(i)})$ uses the stability of $\mathbb{Z}_\Pi^*$ under multiplication by setting

$$c^{(i+1)} = f_a(c^{(i)}) = a\,c^{(i)} \bmod \Pi \quad \text{and} \quad q^{(i+1)} = c^{(i+1)} + \rho \ , \tag{3}$$

where a is a constant appropriately chosen in $\mathbb{Z}_\Pi^*$. We call $\mathsf{G}[n]$ the corresponding algorithm as illustrated in Fig. 5.

The produced *search sequence* $\{q^{(1)}, q^{(2)}, \ldots, q^{(d)}\}$ ends when $q^{(d)} \in \mathcal{P}_c$. Naturally, one has to make sure that the order of f_a (seen as a permutation over $\mathbb{Z}_\Pi^*$) is large enough, that is, a's order in $\mathbb{Z}_\Pi^*$ must be sufficiently large (since $c^{(i+1)} = a^i\,c^{(1)} \bmod \Pi$): otherwise the search sequence could possibly reach a cyclic set of values without ending.

By denoting $\sigma(n, a)$ and $\tau(n)$ the complexity of f_a and T respectively, and $\mathsf{Comp}(n)$ the average time complexity of $\mathsf{G}[n]$, it can be shown that

$$\mathsf{Comp}(n) = \gamma(n) + (\bar{d} - 1)\,\sigma(n, a) + \bar{d}\,\tau(n) \ , \tag{4}$$

where $\bar{d}$ denotes the average sequence length over many trials. Making the heuristic approximation that the random variables induced by the $q^{(i)}$s are independent and uniformly distributed, we get

$$\bar{d} = \frac{\#\mathcal{P}_0}{\#\mathcal{P}_c} \propto \frac{\phi(\eta)}{\eta} \ . \tag{5}$$

It can be shown that our heuristic algorithm $\mathsf{G}[n]$ outputs random n-bit primes in average time complexity $\mathrm{O}(n^4/\log n)$, although we do not give a proof of this fact here due to the lack of space.

From a practical viewpoint, since g and T are given, the only remaining degree of freedom resides in f_a. Note that $\sigma(n,a)$ is multiplied by a potentially big factor, $\#\mathcal{P}_0/\#\mathcal{P}_c$ in (4), so that decreasing $\sigma(n,a)$ leads to a proportional gain in $\mathsf{Comp}(n)$.

We now specialize Eq. (3) so that the transition step is very fast: the best possible value is $a = 2$. In this respect, we exclude $p_1 = 2$ from Π's factorization. The benefit is immediate due to the fact that for all $c \in \mathbb{Z}_\Pi$, $f_2(c) = 2c \bmod \Pi$ only requires *non-modular* additions: $f_2(c) = 2c$ or $2c - \Pi$. Note here that, since Π is odd, $f_2(c)$ can be odd or even. Hence from Eq. (3), our candidate for primality $q = c + \rho$ can be even! So, in order to avoid useless tests, we suggest the following modification: we define Π as $\Pi = (\varepsilon_{\max} - 1)\eta$ and ρ as in Eq. (2). Next, $q = c + \rho$ is optionally added to η according to its parity so that the resulting q is always odd. Here is the final algorithm. We give practical values for η, Π and ρ to generate 512-bit primes in the next section.

Precomputations: parameters η, $\Pi = (\varepsilon_{\max} - 1)\eta$, and $\rho = \varepsilon_{\min}\,\eta$
Output: a n-bit prime q

1. $c = \mathsf{g}()$
2. $q = c + \rho$
3. if q is even then $q \leftarrow q + \eta$
4. if $\mathsf{T}(q) = \texttt{false}$ then $c \leftarrow 2c \bmod \Pi$ and goto 2
5. output q

Fig. 6. $\mathsf{GPrime}[n]$ – An optimized prime number generator.

Alternatively, one may use an even Π and fix a to some particular invertible value modulo Π so that multiplying by a requires very few operations (e.g., $a = 2^{16} + 1$).

6 Implementation Results

After having implemented g on Infineon's SLE66CX160S smart-card platform (8-bit CPU and 1100-bit arithmetic crypto-processor) for $n = 512$ and

$$\begin{aligned}
\eta &= \texttt{b16bd1e084af628fe5089e6dabd16b5b80f60681d6a092fc} \\
&\quad\ \texttt{b1e86d82876ed71921000bcfdd063fb90f81dfd} \\
&\quad\ \texttt{07a021af23c735d52e63bd1cb59c93cbb398afd}_{16}\ , \\
\Pi &= 1729 \cdot \eta\ , \\
\rho &= 4180 \cdot \eta\ ,
\end{aligned}$$

we compute a uniformly distributed[5] random invertible number modulo Π in less than 40 ms at 3.57 MHz. Algorithm on Fig. 5 with $a = 2$ runs in about 3.150 seconds in average to generate a 512-bit prime, which is in high accordance with Eq. (4) (T is a basic Fermat test with base 2 running in $\tau(512) \approx 90$ ms). As a direct consequence, this particularly fast smart-card implementation allows 1024-bit RSA keys on-board generation in less than 8 seconds in average. The generation of an invertible number using g requires about 2.7 KB of code memory (due to the storage of the θ_i). Such a large memory consumption can be avoided by replacing g with g', which only implies the storage of Π and

$$\lambda(\Pi) = \texttt{1dc6c203d4cc780033f9c5d8d97aa2468a54e3700}_{16} \ .$$

This implementation choice has a little impact on performances since the whole RSA key generation process still runs in less than 10 seconds in average. As a comparison with classical methods, we give on Fig. 6 the (heuristic) expected number of calls to T needed by $\mathsf{G}[n]$ and $\mathsf{H}[n, 10]$.

n	256	384	512	640	768	896	1024
$\mathsf{G}[n]$	18.72	26.12	33.29	40.25	46.90	53.56	59.98
$\mathsf{H}[n, 10]$	28.03	42.04	56.05	70.07	84.08	98.1	112.1

Fig. 7. $\mathsf{G}[n]$ *vs* $\mathsf{H}[n, 10]$ – Heuristic expected number of calls to the primality oracle T.

7 Applications

We now apply the previously analyzed tools to some particular contexts. We believe that these techniques constitute a serious improvement on current prime number generators in almost every circumstances, including while implementing ANSI X9.31 recommendations.

7.1 Generation of DSA primes

Here we focus on the problem of generating a uniformly distributed random n-bit prime $p = 1 + qr$ for a given 160-bit prime q. Trial divisions are intended to check that the candidate p has no prime factor p_i for $i = 1, \ldots, k$. As before, we can advantageously generate r so that p automatically fulfills this condition. It suffices that

$$p \not\equiv 0 \pmod{p_i} \iff r \not\equiv -\frac{1}{q} \pmod{p_i} \qquad \text{for } i = 1, \ldots, k \ . \tag{6}$$

[5] Again, we consider the bias of Section 4 to be negligible.

Choosing $\Pi = p_1^{\delta_1} \cdots p_k^{\delta_k}$ with $|\Pi| = |r| = n - 160$, Eq. (6) can be rewritten as

$$r = -\frac{1}{q} + c \bmod \Pi \tag{7}$$

where $c \in \mathbb{Z}_\Pi^*$.

Based on Fig. 5, we therefore propose algorithm $\mathsf{GDSA}[n, q]$. Again, as in Section 5, $\mathsf{g}()$ generates elements of $\mathbb{Z}_\Pi^*$ and $f_a(c) = ac \bmod \Pi$ for some $a \in \mathbb{Z}_\Pi^*$.

Input: a 160-bit prime q
Output: a n-bit prime satisfying $p = 1 + rq$

1. compute $1/q = q^{\lambda(\Pi)-1} \bmod \Pi$
2. $c = \mathsf{g}()$
3. $r = (-1/q + c) \bmod \Pi$
4. $p = 1 + qr$
5. if $\mathsf{T}(p) = \texttt{false}$ then $c \leftarrow f_a(c)$ and goto 3
6. output p

Fig. 8. $\mathsf{GDSA}[n, q]$ – DSA prime generation algorithm based on g.

As a comparison with classical methods, we also give benchmarks for $\mathsf{GDSA}[n, q]$ and $\mathsf{H}[n, 10]$.

n	256	384	512	640	768	896	1024
$\mathsf{GDSA}[n, q]$	22.37	28.71	35.34	42.06	48.62	55.12	61.6
$\mathsf{H}[n, 10]$	28.03	42.04	56.05	70.07	84.08	98.1	112.1

Fig. 9. $\mathsf{GDSA}[n, q]$ – Heuristic expected number of calls to T.

7.2 Generation of safe primes

A prime p is said to be safe if $p = 1 + 2q$ where q is also prime. In order to generate a safe n-bit prime $p = 1 + 2q$, we have to produce a search sequence of pairs $(p^{(i)}, q^{(i)})$ in which $p^{(i)} = 1 + 2q^{(i)}$ and $p^{(i)}$, $q^{(i)}$ are both invertible modulo Π. This can be done by finding for $\Pi = p_1^{\delta_1} \cdots p_k^{\delta_k}$ a value close to 2^{n-2} with maximum k. As we know how to generate an element c of $\mathbb{Z}_\Pi^*$, we propose to test $q^{(1)} = c + \Pi$ and $p^{(1)} = 1 + 2c + 2\Pi$ for primality. By construction, since $c \in \mathbb{Z}_\Pi^*$, $q^{(1)}$ is indeed co-prime to Π and thus makes a good candidate for being a prime: this is however not the case for $p^{(1)}$. For solving this drawback, we propose to modify g into g_s as given in Fig. 10.

Precomputations: $\Pi = \prod_1^k p_i^{\delta_i}$, $t = C \cdot \max_i |p_i^{\delta_i}|$ $(C = 2)$, $\{\theta_i\}$
Input: a random sequence α
Output: a uniformly distributed invertible number $c \in \mathbb{Z}_\Pi^*$ with $1 + 2c \in \mathbb{Z}_\Pi^*$

1. $c = 0$
2. for $i = 1$ to k
   2.1 pick a t-bit random number α_i
   2.2 if $\alpha_i^{\delta_i}\theta_i \bmod \Pi = 0$ goto 2.1
   2.3 if $(1 + 2\alpha_i)^{\delta_i}\theta_i \bmod \Pi = 0$ goto 2.1
   2.4 $c \leftarrow c + \alpha_i\theta_i \bmod \Pi$
3. output c

Fig. 10. Generator $\mathtt{g}_s$ for safe prime generation.

Ouput: a safe n-bit prime $p = 1 + 2q$ with q prime

1. $c = \mathtt{g}_s()$
2. $q = c + \Pi$
3. $p = 1 + 2q$
4. if $\mathsf{T}(p) = \mathtt{false}$ or $\mathsf{T}(q) = \mathtt{false}$ goto 1
5. output p

Fig. 11. $\mathsf{Gsafe}[n]$ – safe prime generation algorithm.

From this modified generator, we naturally define an algorithm $\mathsf{Gsafe}[n]$ generating safe primes as shown on Fig. 11. Since it appears uneasy to find a low-cost transformation $(p^{(i+1)}, q^{(i+1)}) = f(p^{(i)}, q^{(i)})$ that respects co-primality to Π, $\mathtt{g}_s$ is recalled as many times as necessary.

7.3 Application to ANSI X9.31

In order to thwart certain classes of attacks on RSA, the ANSI recommends the use of prime factors satisfying particular properties as exposed in the specifications of X9.31. According to the standard, each prime factor q must be chosen such that

$$\begin{cases} q - 1 \text{ has a large prime divisor } u, \\ q + 1 \text{ has a large prime divisor } s, \end{cases}$$

where the respective sizes of u and s are chosen close to 100 bits. Primes numbers featuring this property will be called X9.31-compliant primes. We first note that, after having chosen parameters $\eta \approx 2^{99}$ and $\Pi = \rho = \eta$, our algorithm $\mathsf{G}[100]$ outputs two 100-bit prime numbers u and s with an expected complexity of 8.73 primality tests. We still have to generate a n-bit prime q such that

$$q = 1 + r_1 \cdot u = -1 + r_2 \cdot s \ ,$$

where r_1 and r_2 are integers of bit-size $n-100$. Hence $r_1 \equiv -\frac{2}{u} \pmod{s}$ and there must be an integer r_3 such that

$$q = 1 + u \cdot (-\frac{2}{u} \bmod s + r_3 \cdot s) \; .$$

By a reasoning similar to the one of Section 7.1, we are driven to produce candidates q of the preceding form with

$$r_3 = -\frac{1}{su} - \frac{-2u^{-1} \mod s}{s} + c \mod \Pi \; ,$$

where $c \in \mathbb{Z}_\Pi^*$ and Π is a product of small primes of total size close to $n-200$. Note also that the intermediate computations

$$\kappa = 1 + u(-2u^{-1} \mod s)$$

and

$$\mu = -\kappa(su)^{-1} \mod \Pi$$

of respective bit-sizes 200 and $n-200$ can be done easily in two exponentiations

$$u^{-1} = u^{s-2} \mod s$$

and

$$(su)^{-1} = (su)^{\lambda(\Pi)-1} \mod \Pi \; .$$

This motivates algorithm $\mathsf{Gx9.31}[n]$ illustrated on Fig. 12. As before, a is a constant chosen in $\mathbb{Z}_\Pi^*$ and $f_a(c) = ac \mod \Pi$. We also give the expected number of calls to the primality oracle T as a function of n on Fig. 13.

Precomputations: Π and $\lambda(\Pi)$
Output: a X9.31-compliant n-bit prime q

1. generate u and s using $\mathsf{G}[100]$
2. compute $\kappa \leftarrow 1 + u \cdot (-2u^{-1} \bmod s)$ and $\mu \leftarrow -\kappa(su)^{-1} \bmod \Pi$
3. $c \leftarrow \mathsf{g}()$
4. $r \leftarrow \mu + c \bmod \Pi$ and $q \leftarrow \kappa + su \cdot r$
5. if $\mathsf{T}(q) = \texttt{false}$ then $c \leftarrow f_a(c)$ and goto 4
6. output q

Fig. 12. $\mathsf{Gx9.31}[n]$ – X9.31-compliant prime generation algorithm.

n	256	384	512	640	768	896	1024
G×9.31$[n]$	25.15	29.64	36.05	42.68	49.18	55.54	61.90

Fig. 13. G×9.31$[n]$ – Heuristic expected number of calls to the primality oracle T.

7.4 Generation of strong primes

A prime number q is said to be *strong* when

$$\begin{cases} q-1 \text{ has a large prime divisor } u, \\ q+1 \text{ has a large prime divisor } s, \\ u-1 \text{ has a large prime divisor } t. \end{cases}$$

The property of being strong therefore implies $X9.31$-compliance. Usually, the bit-sizes of u, s and t are chosen fixed to constant values and hence do not depend on the bit-size of q, n. We will take $|s| = |t| = 100$ and $|u| = 130$ here as an illustrative example, despite the fact that our technique remains fully generic towards these parameters.

Clearly, we can take advantage of the algorithm G×9.31$[n]$ of the preceding section and include the additional stage $u = \mathsf{GDSA}[130, t]$ before the search sequence takes place. This can be done by setting $\Pi \approx 2^{29}$ in $\mathsf{GDSA}[130, t]$. This gives the algorithm Gstrong described on Fig. 14.

Precomputations: Π and $\lambda(\Pi)$
Output: a n-bit strong prime q

1. generate s and t using G[100]
   generate u using $\mathsf{GDSA}[130, t]$
2. compute $\kappa \leftarrow 1 + u \cdot (-2u^{-1} \mod s)$ and
   $\mu \leftarrow -\kappa(su)^{-1} \mod \Pi$
3. $c \leftarrow \mathsf{g}()$
4. $r \leftarrow \mu + c \mod \Pi$ and $q \leftarrow \kappa + su \cdot r$
5. if $\mathsf{T}(q) =$ `false` then $c \leftarrow f_a(c)$ and goto 4
6. output q

Fig. 14. Gstrong$[n]$ – A strong prime generation algorithm.

We stress the fact that our technique features a dramatic performance improvement compared to classical methods. To illustrate this, we give a comparison of the average number of calls to T executed by Gstrong and the classical method, Gordon's algorithm.

n	256	384	512	640	768	896	1024
Gstrong$[n]$	30.34	30.82	36.7	43.1	49.55	56	62.21
Gordon	88.73	133.1	177.45	221.8	266.17	310.53	354.9

Fig. 15. Gstrong$[n]$ *vs* Gordon – Heuristic expected number of calls to the primality oracle T.

7.5 Application in a shared RSA protocol

In [3], Boneh and Franklin proposed a shared RSA protocol which enables two parties with the help of a third party to generate a shared RSA key $N = pq$ and $de \equiv 1 \pmod{\phi(N)}$. In this protocol, N and e are public but p, q and d are shared through a secret sharing algorithm.

The Boneh-Franklin protocol enables the two parties to decrypt. One crucial step in this protocol resides in the protocol generating N. Basically, both parties A and B choose (p_A, q_A) and (p_B, q_B) respectively, proceed to a protocol such that they share $N = (p_A + p_B)(q_A + q_B)$, and check that $p = p_A + p_B$ and $q = q_A + q_B$ are simultaneously prime. Prior to this protocol, A and B check whether p and q are not divisible by small primes. In other words, they first generate some shared p and q which have no small prime factors $p_1, \ldots, p_m$ and start again until p and q are both prime. This leads to an expected number of $\left(\frac{e^{-\gamma}\log 2n}{\log p_m}\right)^2$ joint primality tests. As an example, Boneh and Franklin proposed for $n = 512$ that m should be close to 1024 which leads to a number of trial division steps of 32. Alternatively we propose to generate p and q as

$$p = (p'_A p'_B + (p''_A + p''_B)\Pi) \bmod \varepsilon\Pi \quad \text{and} \quad q = (q'_A q'_B + (q''_A + q''_B)\Pi) \bmod \varepsilon\Pi$$

where $\Pi = \prod_{i=1}^{k} p_i$, $\varepsilon\Pi \approx 2^n$ and p'_A, p'_B, q'_A and q'_B are random numbers co-prime to Π generated by generator g, and then to perform trial divisions by $p_{k+1}, \ldots, p_m$. For $m = 1024$ and $k = 74$, letting $\chi = \prod_{i=75}^{1024} p_i$, the number of trials is then $\chi / \phi(\chi)$ which is approximately 3 instead of 32. This drastically reduces the number of exchanged values in the protocol.

8 Conclusion

We introduced new algorithms for generating pseudo-random numbers with no small factors, and showed how to use them in designing prime number generation algorithms to improve related problems. We gave a sketchy expression of our main algorithm's complexity in heuristic terms: this complexity relates to the distribution of prime numbers in the arithmetic progression $a^i c \mod \Pi + \rho$ with $i \geq 0$ and $a, c \in \mathbb{Z}_\pi^*$. Therefore, an open question would be to provide a more formal investigation on the distribution of those primes, the same way Brandt and Damgård [5] characterized the naive incremental generator.

Acknowledgements

We are grateful to the referees for their useful comments.

References

1. ANSI X9.31. Public-key cryptography using RSA for the financial services industry. American National Standard for Financial Services, draft, 1995.
2. A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. *Mathematics of Computation*, vol. 61, pp. 29–68, 1993.
3. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Advances in Cryptology – CRYPTO '97*, vol. 1294 of Lecture Notes in Computer Science, pp. 425–439, Springer-Verlag, 1997.
4. W. Bosma and M.-P. van der Hulst. Faster primality testing. In *Advances in Cryptology – CRYPTO '89*, vol. 435 of Lecture Notes in Computer Science, pp. 652–656, Springer-Verlag, 1990.
5. J. Brandt and I. Damgård. On generation of probable primes by incremental search. In *Advances in Cryptology – CRYPTO '92*, vol. 740 of Lecture Notes in Computer Science, pp. 358–370, Springer-Verlag, 1993.
6. J. Brandt, I. Damgård, and P. Landrock. Speeding up prime number generation. In *Advances in Cryptology – ASIACRYPT '91*, vol. 739 of Lecture Notes in Computer Science, pp. 440–449, Springer-Verlag, 1991.
7. C. Couvreur and J.-J. Quisquater. An introduction to fast generation of large prime numbers. *Philips Journal of Research*, vol. 37, pp. 231–264, 1982.
8. C. Ding, D. Pei, and A. Salomaa. *Chinese Remainder Theorem*, Word Scientific, 1996.
9. FIPS 186. Digital signature standard. Federal Information Processing Standards Publication 186, US Department of Commerce/N.I.S.T., 1994.
10. D.E. Knuth. *The Art of Computer Programming - Seminumerical Algorithms*, vol. 2, Addison-Wesley, 2nd ed., 1981.
11. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.
12. H.C. Pocklington. The determination of the prime or composite nature of large numbers by Fermat's theorem. *Proc. of the Cambridge Philosophical Society*, vol. 18, pp. 29–30, 1914.
13. H. Riesel. *Prime Numbers and Computer Methods for Factorization*, Birkhäuser, 1985.
14. R.L. Rivest. Remarks on a proposed cryptanalytic attack on the M.I.T. public-key cryptosystem. *Cryptologia*, vol. 2, pp. 62–65, 1978.
15. R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, vol. 21, pp. 120–126, 1978.
16. R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, vol. 6, pp. 84–85, 1977.

Chinese Remaindering Based Cryptosystems in the Presence of Faults

[Published in *Journal of Cryptology* **12**(4):241–245, 1999.]

Marc Joye[1], Arjen K. Lenstra[2], and Jean-Jacques Quisquater[3]

[1] UCL Crypto Group, Dép. de Mathématique, Université de Louvain
Chemin du Cyclotron, 2, B-1348 Louvain-la-Neuve, Belgium
mjoye@geocities.com
[2] Citibank, N.A.
4 Sylvan Way, Parsippany, NJ 07054, U.S.A.
arjen.lenstra@citicorp.com
[3] UCL Crypto Group, Lab. de Microélectronique, Université de Louvain
Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium
jjq@dice.ucl.ac.be

Abstract. We present some observations on public-key cryptosystems that use the Chinese remaindering algorithm. Our results imply that careless implementations of such systems could be vulnerable. Only one faulty signature, in some explained context, is enough to recover the secret key.

Keywords. Public-key cryptosystems, faulty computations, Chinese remaindering.

1 Introduction

In public-key cryptosystems two distinct computations can be distinguished: the computation that makes use of the secret, public key pair, and the one that only makes use of the public key. The former usually corresponds to the secret decryption or to the signature generation operation, the latter to the public encryption or to the signature verification operation. In this paper we restrict our attention to public key cryptosystems in which the former computation can be sped up using the Chinese remaindering algorithm. Examples of such cryptosystems are: RSA [16], LUC [19], KMOV [11], and Demytko's cryptosystem [6]. We show that devices implementing the signature generation of any of these cryptosystems may be tricked into revealing their secret key, if the following three conditions are met:

(1) the message as signed is known;
(2) a certain type of faulty behavior occurs during signature generation;
(3) the device outputs the faulty signature.

Leakage of the secret key can be averted by making sure that either of these three conditions will not be met. This can be done by adding enough random noise to the message to be signed, by making sure that the system works properly and that faulty behavior cannot be induced, by checking the correctness of the signature before outputting it, or by any combination of these three safety measures. We note that many devices already implement at least one of these countermeasures. Thus we feel confident that the practical impact of our observation is minimal. Therefore, if carefully implemented, Chinese remaindering based cryptosystems are not more vulnerable than usual cryptosystems.

We note that the same observation applies to decryption using the Chinese remaindering algorithm, if the decrypting party shows the faulty decryption to another party. The details of this 'generalization' are straightforward.

The analysis of cryptosystems in the presence of faults was launched by newspaper publications that cited a Bellcore press release *New Threat Model Breaks Crypto Codes*. Thereafter, several researchers reported some possible implications in both public-key [12], [3], [8], [9], [17], [20] and private-key [4], [9], [13] cryptography. The method presented in this paper improves the Bellcore's result, later published in [5], in the following way. Their method requires two 'Chinese remaindering' signatures on the same message, one correct and one faulty, whereas our version requires the message and only a single faulty signature. Our version is therefore 'more realistic' and potentially more dangerous.

The problem of the presence of faults in cryptosystems can be turned into an active attack by inducing faulty behavior on computational devices. For example, this can be achieved by ROM overwriting, EEPROM modification, gate destruction, RAM remanence, etc... [1], [2], [7], [10], [14], [15]. Since these techniques are not fully published or described (and thus controversial), we do not elaborate or comment.

Our main objective is to dwell on the importance of a careful implementation of cryptosystems. Suppose you are in a context involving Trusted Third Parties (e.g., banks) and where thousands of signatures are produced each day. If, for some reason or other, a single signature is faulty, then the security of the whole system may be compromised.

2 Potential Vulnerability of RSA Using Chinese Remaindering

Let p and q be two primes and let $n = pq$. Imagine a message m is signed with the secret exponent d using RSA: $s = m^d \bmod n$. Using the Chinese remaindering theorem, the value of s can be computed more efficiently from $s_p = m^d \bmod p$ and $s_q = m^d \bmod q$. Suppose an error occurs during the computation of s_p (we denote s'_p the faulty value), but not during the computation of s_q. Applying Chinese remaindering on s'_p ($\neq s_p$) and s_q will give the faulty signature s' for message m. Then, the computation of

$$\gcd(s'^e - m \pmod n, n)$$

will give the secret factor q, where e is the public exponent.

Remarks. 1) If the attacker does not know the public modulus n, he may still be able to recover the secret parameter q. Indeed, if e is small, he has some probability to find q by trying to factorize ${s'}^e - m$ over the rational integers. This probability becomes non negligible if he possesses two or several faulty signatures, because in that case, he can recover q by computing $\gcd({s'_1}^e - m_1, \ldots, {s'_k}^e - m_k)$. Furthermore, from the knowledge of one or several valid signatures, the attacker can also recover p in a similar way.
2) A non-trivial factor of n may be derived in any scenario were exactly one of the remainders used in the Chinese remaindering is incorrect. This includes, for instance, incorrect retrieval of (a possibly correctly computed) s_p. Thus, also the presence of a permanent failure, like a damaged wire, as opposed to a transient computational failure, may expose secret information.

3 Generalization to Other Cryptosystems

In this section, $n = pq$ denotes the RSA-modulus, and e and d are respectively the public and the secret exponents. The message to be signed is m, and the corresponding signature is s. Let

$$\mathcal{S} : \mathbb{Z}_n \to \mathbb{Z}_n, m \mapsto s = \mathcal{S}(m)$$

be an RSA-type signature function. If the signature s of message m is computed with the Chinese remaindering theorem, then the previous observation still applies.

Proposition 1. *If s' is a faulty signature such that $s' \not\equiv s \pmod{p}$ but $s' \equiv s \pmod{q}$, then*

$$\gcd(\mathcal{S}^{-1}(s') - m, n)$$

will give the secret factor q.

Proof. Since $s' \equiv s \pmod{q}$ and $s' \not\equiv s \pmod{p}$, we have $\mathcal{S}^{-1}(s') \equiv \mathcal{S}^{-1}(s) \equiv m \pmod{q}$ and $\mathcal{S}^{-1}(s') \not\equiv m \pmod{p}$. Hence, $\mathcal{S}^{-1}(s') - m \pmod{n}$ is divisible by q and not by p. □

Consequently, the observation of Section 2 works for all RSA-type cryptosystems.

Example 1. The LUC cryptosystem is based on Lucas sequences. The signature function is defined as $\mathcal{S}(m) = V_d(m, 1) \bmod n$, and the verification function as $\mathcal{S}^{-1}(s) = V_e(m, 1) \bmod n$, where $ed \equiv 1 \pmod{\operatorname{lcm}(p-1, p+1, q-1, q+1)}$. If $s' \not\equiv s \pmod{p}$ but $s' \equiv s \pmod{q}$, then

$$\gcd(V_e(s', 1) - m \pmod{n}, n)$$

will give q.

Example 2. The Demytko cryptosystem uses the x-coordinate of points on elliptic curves over the ring $\mathbb{Z}_n$. Such a curve will be denoted by $E_n(a,b)$. The x-coordinate of the multiple of a point can be computed thanks to the division polynomials (see [18, exercice 3.7]) considered as polynomials in $\mathbb{Z}_n[a,b,x]$. The signature function is defined as $\mathcal{S}(m) = \Phi_d(m)/\Psi_d(m)^2 \bmod n$, and the verification function as $\mathcal{S}^{-1}(s) = \Phi_e(s)/\Psi_e(s)^2 \bmod n$, where $ed \equiv 1 \pmod{\operatorname{lcm}\big(\#E_p(a,b), \#E_q(a,b), \#\overline{E_p(a,b)}, \#\overline{E_q(a,b)}\big)}$.[1] If $s' \not\equiv s \pmod p$ but $s' \equiv s \pmod q$, then

$$\gcd\left(\frac{\Phi_e(s')}{\Psi_e(s')^2} - m \pmod n, n\right)$$

will give q.

Acknowledgements

We thank François Koeune for some useful comments.

References

1. R. Anderson and M. Kuhn. Tamper resistance—a cautionary note. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, pp. 1–11. USENIX Association, Berkeley, 1996.
2. R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In B. Christianson, B. Crispo, M. Lomas, and M. Roe, editors, *Security Protocols*, pp. 125–136. Lecture Notes in Computer Science, vol. 1361. Springer-Verlag, Berlin, 1998.
3. F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimhalu, and T. Ngair. Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. In B. Christianson, B. Crispo, M. Lomas, and M. Roe, editors, *Security Protocols*, pp. 115–124. Lecture Notes in Computer Science, vol. 1361. Springer-Verlag, Berlin, 1998.
4. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B. S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97*, pp. 513–525. Lecture Notes in Computer Science, vol. 1294. Springer-Verlag, Berlin, 1997.
5. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT '97*, pp. 37–51. Lecture Notes in Computer Science, vol. 1233. Springer-Verlag, Berlin, 1997.
6. N. Demytko. A new elliptic curve based analogue of RSA. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, pp. 40–49. Lecture Notes in Computer Science, vol. 765. Springer-Verlag, Berlin, 1994.
7. P. Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Proceedings of Sixth USENIX Security Symposium*, pp. 77–89. USENIX Association, Berkeley, 1996.

[1] $\overline{E_p(a,b)}$ denotes the complementary group of $E_p(a,b)$. See the original paper [6] for a detailed description.

8. M. Joye and J.-J. Quisquater. Attacks on systems using Chinese remaindering. Technical Report CG-1996/9, UCL Crypto Group, Louvain-la-Neuve, November 1996.
9. B. S. Kaliski Jr and M. J. B. Robshaw. Comments on some attacks on cryptographic devices. RSA Laboratories' Bulletin, no. 5, RSA Laboratories, Redwood City, July 1997.
10. O. Kocar. Hardwaresicherheit von Mikrochips in Chipkarten. *Datenschutz und Datensicherheit*, vol. 20, no. 7, pp. 421–424, July 1996.
11. K. Koyama, U. M. Maurer, T. Okamoto, and S. A. Vanstone. New public-key schemes based on elliptic curves over the ring $\mathbb{Z}_n$. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, pp. 252–266. Lecture Notes in Computer Science, vol. 576. Springer-Verlag, Berlin, 1992.
12. A. K. Lenstra. RSA signature generation in the presence of faults. Memo, Parsippany, September 1996.
13. P. Paillier. Real life DFA is inherently limited. Presented at the rump session of EUROCRYPT '97, 11–15th May 1997, Konstanz.
14. I. Peterson. Chinks in digital armor—Exploiting faults to break smart-card cryptosystems. *Science News*, vol. 151, no. 5, pp. 78–79, February 1997.
15. J.-J. Quisquater. The adolescence of smart cards. *Future Generation Computer Systems*, vol. 13, no. 1, pp. 3–7, June 1997.
16. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.
17. A. Shamir. How to check modular exponentiation. Presented at the rump session of EUROCRYPT '97, 11–15th May 1997, Konstanz.
18. J. H. Silverman. *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, vol. 106. Springer-Verlag, New York, 1986.
19. P. J. Smith and M. J. J. Lennon. LUC: a new public key system. In E. G. Douglas, editor, *Proceedings of the Ninth IFIP Symposium on Computer Security*, pp. 103–117. Elsevier, Amsterdam, 1993.
20. Y. Zheng and T. Matsumoto. Breaking real-world implementations of cryptosystems by manipulating their random number generation. In *Pre-proceedings of the 1997 Symposium on Cryptography and Information Security*, 29th January–1st February 1997, Fukuoka.

Universal Exponentiation Algorithm
– A First Step Towards *Provable* SPA-resistance –

[Published in Ç.K. Koç, D. Naccache, and C. Paar, Eds., *Cryptographic Hardware and Embedded Systems – CHES 2001*, vol. 2162 of *Lecture Notes in Computer Science*, pp. 300–308, Springer-Verlag, 2001.]

Christophe Clavier and Marc Joye

Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
{christophe.clavier, marc.joye}@gemplus.com
http://www.geocities.com/MarcJoye/

Abstract. Very few countermeasures are known to protect an exponentiation against simple side-channel analyses. Moreover, all of them are heuristic.
This paper presents a universal exponentiation algorithm. By tying the exponent to a corresponding addition chain, our algorithm can virtually execute any exponentiation method.
Our aim is to transfer the security of the exponentiation method being implemented to the exponent itself. As a result, we hopefully tend to reconcile the provable security notions of modern cryptography with real-world implementations of exponentiation-based cryptosystems.

Keywords. Implementation, exponentiation, RSA cryptosystem, discrete logarithm, side-channel attacks, simple power analysis (SPA), provable security, addition chains, smart cards.

1 Introduction

The security of a cryptosystem is evaluated as the latter's ability to resist attacks in a given adversarial model. It is very challenging to guess the strategy the adversary will follow in an attempt to break the system. So, the only assumptions made by modern cryptography refer to the *computational abilities* of the adversary [6]. Loosely speaking, a cryptosystem is then said *secure* if there is no polynomial-time adversary able to gain more "useful" information than a honest user by deviating from the "prescribed" behavior.

In [9, 11], Kocher *et al.* launched a new class of attacks: the so-called *side-channel attacks*. In such a scenario, an adversary monitors some side-channel information (e.g., power consumption) during the execution of a crypto-algorithm and thereby may foil the security of the corresponding "provably secure" cryptosystem. So what does provable security mean? The security is usually proven

by reduction: one shows that the only way to break the cryptosystem is to break the underlying cryptographic primitive (e.g., the RSA function). Since this is assumed to be computationally infeasible, the cryptosystem is declared secure. A side-channel attack does *not* violate this assumption, it just considers other directions to break the cryptographic primitive. Consequently, we stress that the notions of provable security, or more exactly *provable computational security*, are very useful and must be part of the analysis of any cryptosystem.

Unfortunately, there is no counterpart to side-channel attacks. Defining a security model for this class of attacks seems unrealistic since we do not see how to limit the power of the adversary. The best we can hope to prove is the security relative to one particular attack.

This paper focuses on modular exponentiation (e.g., the RSA function or the discrete logarithm function) as a cryptographic primitive. Using a representation with addition chains, we "transfer" the security of the exponentiation method actually implemented in the exponent itself (which is the secret data). The resulting algorithm, which we call *universal exponentiation algorithm*, works with virtually all exponentiation methods. It simply reads triplets of values $(\gamma(i) : \alpha(i), \beta(i))$, meaning that the content of register $R[\alpha(i)]$ must be multiplied by the content of register $R[\beta(i)]$ and that the result must be written into register $R[\gamma(i)]$. We provide in this way a kind of reduction. Instead of carefully analyzing a specific exponentiation method, the implementor simply verifies that the *atomic* operation $R[\gamma(i)] \leftarrow R[\alpha(i)] \cdot R[\beta(i)]$ does not leak any "useful" information through a given side-channel attack. This methodology is reminiscent of the traditional security proofs. In the traditional case, the security of a cryptographic primitive is conjectured (e.g., inverting the RSA function is infeasible) whereas in our case the security of an atomic operation is assessed through experiments (e.g., I cannot "break" a multiplication by SPA). The main difference is that the security assumption is scrutinized by fewer people and hence is more controversial.

The rest of this paper is organized as follows. The next section recalls the definition of an addition chain. Based on it, we then present our universal exponentiation algorithm. In Section 3, we discuss the merits of our approach from a security viewpoint. Section 4 suggests some modifications to our basic algorithm. Finally, we conclude in Section 5.

2 Universal Exponentiation Algorithm

2.1 Addition chains

We start by a brief introduction to addition chains. For further details, we refer the reader to [8].

Definition 1. *An* addition chain *for a positive integer d is a sequence $\mathcal{C}(d) = \{d^{(0)}, d^{(1)}, \ldots, d^{(\ell)}\}$ satisfying*

1. *$d^{(0)} = 1$, $d^{(\ell)} = d$, and*
2. *for all $1 \leq i \leq \ell$, there exist $j(i), k(i) < i$ such that $d^{(i)} = d^{(j(i))} + d^{(k(i))}$.*

Integer ℓ defines the *length* of chain $\mathcal{C}$. An addition chain is called a *star-chain* if for all $1 \leq i \leq \ell$ there exists $k(i) < i$ such that $d^{(i)} = d^{(i-1)} + d^{(k(i))}$.

A slightly more general notion is that of addition-subtraction chains.

Definition 2. *An* addition-subtraction chain *for an integer d is a sequence $\mathcal{C}(d) = \{d^{(0)}, d^{(1)}, \ldots, d^{(\ell)}\}$ satisfying*

1. *$d^{(0)} = 1$, $d^{(\ell)} = d$, and*
2. *for all $1 \leq i \leq \ell$ there exist $j(i), k(i) < i$ such that $d^{(i)} = \pm d^{(j(i))} \pm d^{(k(i))}$.*

2.2 A universal algorithm

Let $\mathcal{C}(d) = \{d^{(0)}, d^{(1)}, \ldots, d^{(\ell)}\}$ be an addition chain for exponent d. So for all $1 \leq i \leq \ell$, we have $d^{(i)} = d^{(j(i))} + d^{(k(i))}$. This provides an easy means to evaluate $y = x^d$: For $i = 1$ to ℓ compute

$$x^{d^{(i)}} = x^{d^{(j(i))}} \cdot x^{d^{(k(i))}}$$

and then set $y = x^{d^{(\ell)}}$. So, from an addition chain of length ℓ, ℓ multiplications are required to compute y.

Example 1. An addition chain for 5 is $\mathcal{C}(5) = \{1, 2, 3, 5\}$ and so $x^1 = x$, $x^2 = x^1 \cdot x^1$, $x^3 = x^2 \cdot x^1$, and finally $x^5 = x^3 \cdot x^2$.

At step i, $x^{d^{(i)}}$ is evaluated as $x^{d^{(i)}} = x^{d^{(j(i))}} \cdot x^{d^{(k(i))}}$. Assuming that $x^{d^{(j(i))}}$ and $x^{d^{(k(i))}}$ respectively belong to registers $R[\alpha(i)]$ and $R[\beta(i)]$ and that the result, $x^{d^{(i)}}$, is written in register $R[\gamma(i)]$, exponent d can be represented by the *register sequence*

$$\Gamma(d) = \big\{\big(\gamma(i) : \alpha(i), \beta(i)\big)\big\}_{1 \leq i \leq \ell}, \tag{1}$$

meaning that $R[\gamma(i)] = R[\alpha(i)] \cdot R[\beta(i)]$. (By convention, the value $d = 1$ is represented by $\Gamma(1) = \emptyset$.)

From this, we obtain the following exponentiation algorithm (for $d > 1$):

Input: $x, \Gamma(d)$ Output: $y = x^d$
$R[\alpha(1)] \leftarrow x$; $R[\beta(1)] \leftarrow x$ for $i = 1$ to ℓ do $R[\gamma(i)] \leftarrow R[\alpha(i)] \cdot R[\beta(i)]$ return $R[\gamma(\ell)]$

Algorithm 1. Universal exponentiation algorithm.

Note that $R[\alpha(1)]$ and $R[\beta(1)]$ are initialized to x because the second item of each addition chain is always $d^{(1)} = 2$. Note also that one may have $\alpha(1) = \beta(1)$.

For star chains, we have $d^{(i)} = d^{(i-1)} + d^{(k(i))}$. Therefore pairs are sufficient to represent d: $\alpha(i) = \gamma(i-1)$ for all $1 \leq i \leq \ell$ and can be omitted from the representation. Hence, we have the *star register sequence*

$$\Gamma^*(d) = \big\{\big(\gamma(i) : \beta(i)\big)\big\}_{1 \leq i \leq \ell} \ . \tag{2}$$

The corresponding exponentiation algorithm is:

Input: $x, \Gamma^*(d)$
Output: $y = x^d$

$R[\gamma(0)] \leftarrow x$; $R[\beta(1)] \leftarrow x$
for $i = 1$ to ℓ do
  $R[\gamma(i)] \leftarrow R[\gamma(i-1)] \cdot R[\beta(i)]$
return $R[\gamma(\ell)]$

Algorithm 2. Universal star exponentiation algorithm.

3 Towards Provable SPA-resistance

The ultimate goal of smart-card manufacturers is a proof that their implementations are resistant to side-channel analysis. In this paper, we adopt the methodology of modern cryptography towards this goal.

Take for example the encryption scheme RSA-OAEP [3]. The minimal security requirement for an encryption scheme is *one-wayness* (OW). This captures the property that an adversary cannot recover the whole plaintext from a given ciphertext. In some cases, partial information about a plaintext may have disastrous consequences. This notion is captured by *semantic security* or the equivalent notion of *indistinguishability* [7]. Basically, indistinguishability means that the only strategy for an adversary to distinguish between the encryptions of any two plaintexts is to guess at random. The strongest attacks one can imagine (at the protocol level) are the so-called *adaptive chosen-ciphertext attacks* (CCA2). Those attacks consider an active adversary who can obtain the decryption of any ciphertext of her/his choice. From the pair of adversarial goal (IND) and adversarial model (CCA2), we derive the security notion of IND-CCA2. In an IND-CCA2 scenario, an adversary has access to a decryption oracle. S/he first outputs a pair of plaintexts m_0 and m_1. Then, given a challenge ciphertext c_b which is either the encryption of m_0 or m_1, the adversary has to guess with a probability non-negligibly better than $1/2$ if c_b encrypts m_0 or m_1. The attack is called adaptive, if after receiving the challenge c_b, the adversary may still obtain decryptions of chosen ciphertexts, the only restriction being not to probe on c_b.

In [3], Bellare and Rogaway remarkably proved that if an adversary is able to break the IND-CCA2 security of RSA-OAEP then the same adversary is able

to break the OW security of the RSA function, that is, to compute an e^{th} root modulo a large composite number $N = pq$ (where typically p and q are 512-bit primes). Since the latter is assumed infeasible, RSA-OAEP is declared provably secure. We note that their proof only holds in the *random oracle model* [2], i.e., an ideal world where hash functions behave like random functions. To summarize, the security of RSA-OAEP is proven by

1. identifying the security goal and the adversarial model (i.e., IND-CCA2);
2. defining the working hypotheses (i.e., random oracle model);
3. exhibiting a reduction (i.e., breaking the IND-CCA2 of RSA-OAEP $\Rightarrow$ breaking the OW of the RSA function);
4. assuming that the reduced problem is intractable (i.e., inverting RSA is infeasible);
5. deducing the security notion (i.e., IND-CCA2 security of RSA-OAEP in the random oracle model).

The security of RSA-OAEP is at the protocol level. To break the IND-CCA2 security, the adversary has a black-box access to a decryption oracle: s/he knows the input and obtains the corresponding output. In the case of side-channel attacks, the adversary is more powerful: s/he gets access to some internal states of the computation.

So, by monitoring the power consumption of an RSA exponentiation, an attacker is even sometimes able to recover the secret decryption exponent d used in the computation of $y = x^d \bmod N$ and the OW assumption of the RSA function is no longer valid. Suppose for example that the RSA function is naively implemented with the square-and-multiply method. As shown in the next figure, the exponent can then be recovered very easily: a lower consumption level corresponds to a squaring and a higher consumption level corresponds to a multiplication.

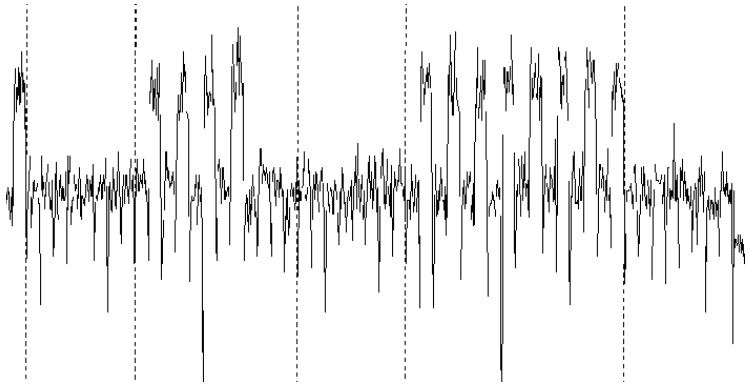


Fig. 1. Power trace of a square-and-multiply exponentiation.

In our simplified model, we consider the fundamental security goal of *unbreakability* (UB). A cryptosystem is said unbreakable if it is infeasible to recover the secret key. This kind of attack is usually referred to as a total breaking. We also consider an attacker who has access to some side-channel information. Depending on the side-channel information and the way it is treated, we define several adversarial models. In the *simple power analysis* (SPA) model, an attacker acquires the power trace of a single execution of the crypto-algorithm. From this, we derive the security notion of UB-SPA. Likewise, one can define the UB-DPA (DPA stands for *differential power analysis* [11]) and so on; one can also consider other security goals and derive security notions like OW-SPA or IND-SPA. It is worth noting here that, contrary to modern cryptography, the definition of an adversarial model is not *absolute*: in a CCA2 attack, an adversary obtains the plaintext corresponding to a chosen ciphertext whereas in an attack like a SPA, the "quality" of the returned information depends on the acquisition tools among other things.

Concentrating on the exponentiation function and more particularly on the RSA function, one can show that if an adversary is able to break the UB-SPA security of the universal exponentiation algorithm, s/he is also able to invert the RSA function. (We note that the main threat for an RSA exponentiation is the SPA; for DPA, efficient counter-measures are known.) In order to break the UB-SPA, an adversary must be able to gain some secret information from the basic operation $R[\gamma(i)] \leftarrow R[\alpha(i)] \cdot R[\beta(i)]$ by SPA, that is, s/he must be able to, at least, differentiate among the triplets $(\gamma(i) : \alpha(i), \beta(i))$ and to recover *all* their values to break the UB property. Assuming that the latter is infeasible (this can be verified experimentally), one has strong evidence[1] that the universal exponentiation algorithm resists to SPA. As a conclusion, if RSA-OAEP is implemented with the universal exponentiation algorithm, we have strong evidence that it resists SPA attacks. Note here that the security is assessed at the implementation level.

From a security viewpoint, the advantage of our method is evident. It reduces the problem of scrutinizing *any* exponentiation algorithm to that of the simpler operation $R[\gamma(i)] \leftarrow R[\alpha(i)] \cdot R[\beta(i)]$. This makes the job of the implementor a lot easier since s/he has a better knowledge of the sensitive parts of her/his algorithm. Moreover, the security passes from a macroscopic level (a software exponentiation) to a microscopic level (a hardware multiplication). Finally, the analysis must be done once for all and remains valid whatever the exponentiation algorithm underlying a given Γ-representation.

Remark 1. In some ways, to relax the assumption the universal exponentiation algorithm is UB-SPA, one can always randomly add dummy operations at the expense of a longer running time (e.g., to add to a Γ representation, a triplet that does not affect the final result). One can also exploit the property that $R[\gamma(i)] \leftarrow R[\alpha(i)] \cdot R[\beta(i)]$ and $R[\gamma(i)] \leftarrow R[\beta(i)] \cdot R[\alpha(i)]$ both lead to the same

[1] In contrast with modern cryptography, we cannot say that we have a proof of security because as aforementioned this depends on the quality of the experiments.

result. Another solution consists to randomly permute the order of the registers and their values during the course of the exponentiation.

In addition to simplifying the security analysis, our universal exponentiation algorithm has the following features:

- it is *simple*: its implementation is straightforward and so programming errors are likely avoided;
- it is *flexible*: owing to the genericity of the Γ-representation, it can virtually execute all exponentiation algorithms;
- it is *fast*: contrary to the protected square-and-multiply method (a.k.a. square-and-multiply-always method) which requires $2\log_2 d$ multiplications for computing $y = x^d$, our algorithm may require as few as $1.25\log_2 d$ multiplications (cf. § 4.1);
- it is *economic*: if the exponentiation algorithm underlying a Γ-representation happens to be flawed, it is enough to correct the Γ-representation: a complete re-programming is unnecessary.

The last property is especially interesting for a smart-card implementation. The program code is usually stored in ROM memory via an expensive process called masking and the secret key (e.g., the RSA decryption exponent d) is stored in EEPROM memory at the personalization stage. So in case of secret leakage or mis-programming, one has just to change or correct the Γ-representation of the secret exponent.

4 Practical Considerations

If we want to realize a smart-card implementation of the proposed algorithms (Algorithms 1 and 2), we face some constraints. A smart-card has a limited number of registers and so we need a way to produce Γ-representations with a predetermined number of registers. Moreover, a Γ-representation with fewer registers requires fewer memory for its storage. Another difficulty may occur when the secret exponent is generated outside the card by a third party because it is given in its binary representation.

In this section, we suggest two different approaches that alleviate the above limitations.

4.1 On-line generation

A straightforward solution is to produce a Γ-representation on-line, i.e., by the smart-card itself. Several good heuristics are known for producing relatively short addition chains. In [12], Walter suggests the following method to compute $y = x^d$ (see also [4]).

Define $d_0 = d$, $x_0 = x$, and $y_0 = 1$. Next, at each step, write $d_i = m_i\, d_{i+1} + r_i$ for appropriately chosen values for (m_i, r_i). Hence, letting $x_{i+1} = {x_i}^{m_i}$ and

$y_{i+1} = {x_i}^{r_i} y_i$, we get

$$\begin{aligned} y = {x_0}^{d_0}\, y_0 &= ({x_0}^{m_0})^{d_1}\, ({x_0}^{r_0}\, y_0) \\ &= {x_1}^{d_1}\, y_1 = ({x_1}^{m_1})^{d_2}\, ({x_1}^{r_1}\, y_1) \\ &= {x_2}^{d_2}\, y_2 = ({x_2}^{m_2})^{d_3}\, ({x_2}^{r_2}\, y_2) \\ &= {x_3}^{d_3}\, y_3 = \cdots \end{aligned}$$

The idea behind Walter's method is to find pairs (m_i, r_i) so that the evaluations of both ${x_i}^{m_i}$ and ${x_i}^{r_i}$ are inexpensive. This is the case when r_i lies in the addition chain used to evaluate ${x_i}^{m_i}$.

Such a method is very well suited to a smart-card implementation. It is easy to implement and the corresponding register sequence, $\Gamma(d)$, requires only one more register than the standard square-and-multiply method. Furthermore, the average length of $\Gamma(d)$ is only $1.25 \log_2 d$, with a very small deviation. See [12] for details.

Note that the computation of $\Gamma(d)$ must be performed in a secured environment since its disclosure reveals the value of secret exponent d. For example, this can performed at the personalization of the card.

4.2 Exponent splitting

The second solution we propose relies on the simple observation that

$$x^d = x^a \cdot x^{d-a} \tag{3}$$

for some a. The idea of splitting the data was already abstracted in [5] as a general countermeasure against differential power analysis attacks. We note that the values of *both* a and $(d-a)$ are required to recover the value of d. In other words, only one exponentiation, x^a or x^{d-a}, needs to be secured.

Given a register sequence for a, $\Gamma(a)$ or $\Gamma^*(a)$, we can compute $y' = x^a$ and $d' = d - a$, and so $x^d = y' \cdot x^{d'}$. There are two possible alternatives. The first one is, for a given a, to store a *chosen* (and thus fixed) register sequence, $\Gamma(a)$, during the personalization of the card. (In this case a star representation, $\Gamma^*(a)$, may be preferred since it requires fewer memory.) The advantage of this approach is that this imposes the underlying methods for computing $y' = x^a$ and $d' = d - a$.

Another alternative consists in *randomly* computing a register sequence, $\Gamma(a)$ or $\Gamma^*(a)$, for a "on the fly". The advantages of this second approach are twofold. First, no register sequence needs to be stored in non-volatile memory and so this results in some memory savings. Second, the methods for evaluating $y' = x^a$ and $d' = d - a$ differ at each execution. Independently, this randomization also helps to prevent differential attacks like the DPA.

5 Conclusion

In this paper, we presented an universal exponentiation algorithm. Through the notion of register sequence, $\Gamma(d) = \{(\gamma(i) : \alpha(i), \beta(i))\}_{1 \le i \le \ell}$, built from

addition chains, we explained how this helps to protect an exponentiation-based cryptosystem against simple side-channel attacks like SPA. Assuming that a more atomic operation (i.e., the multiplication of registers $R[\gamma(i)] \leftarrow R[\alpha(i)] \cdot R[\beta(i)]$) does not leak secret information, we "proved" the security of our implementation. There is no secret at all involved in our universal exponentiation algorithm: the secret exponent d is intimately tied to $\Gamma(d)$ and recovering the value of d supposes the recovery of the whole sequence $\Gamma(d)$, which is a contradiction. Furthermore, our algorithm can be trivially implemented and it greatly simplifies the security analysis since the critical (i.e., sensitive) parts are better understood.

As a final conclusion, we hope that this first step towards provable security of real-world implementations will be a motivating starting-point for further research in this very important subject.

Acknowledgements

The authors are grateful to Jacques Fournier, Karine Gandolfi and Florence Quès for some comments.

References

1. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. Full paper (30 pages), February 1999. An extended abstract appears in H. Krawczyk, ed., Advances in Cryptology – CRYPTO '98, volume 1462 of Lecture Notes in Computer Science, pages 26–45, Springer-Verlag, 1998.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
3. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.
4. F. Bergeron, J. Berstel, S. Brlek, and C. Duboc. Addition chains using continued fractions. *Journal of Algorithms*, 10(3):403–412, September 1989.
5. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, 1999.
6. Oded Goldreich. On the foundations of modern cryptography. In B. Kaliski, editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 46–74. Springer-Verlag, 1997.
7. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
8. Donald E. Knuth. *The art of computer programming/Seminumerical algorithms*, volume 2. Addison-Wesley, 2nd edition, 1981.
9. Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

10. Paul Kocher. Secure modular exponentiation with leak minimization for smart cards and other cryptosystems. International patent WO 99/67909, March 1998.
11. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
12. Colin D. Walter. Exponentiation using division chains. *IEEE Transactions on Computers*, 47(7):757–765, July 1998.
13. Yacov Yacobi. Exponentiating faster with addition chains. In *Advances in Cryptology – EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 222–229. Springer-Verlag, 1991.

Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity

[To appear in *IEEE Transactions on Computers*.]

Benoît Chevallier-Mames[1], Mathieu Ciet[2], and Marc Joye[1,⋆]

[1] Gemplus, Card Security Group
La Vigie, Avenue du Jujubier, ZI Athélia IV, 13705 La Ciotat Cedex, France
{benoit.chevallier-mames,marc.joye}@gemplus.com
http://www.gemplus.com/smart/
[2] UCL Crypto Group
Place du Levant 3, 1348 Louvain-la-Neuve, Belgium
ciet@dice.ucl.ac.be − http://www.dice.ucl.ac.be/crypto/

Abstract. This paper introduces simple methods to convert a cryptographic algorithm into an algorithm protected against simple side-channel attacks. Contrary to previously known solutions, the proposed techniques are not at the expense of the execution time. Moreover, they are generic and apply to virtually any algorithm.
In particular, we present several novel exponentiation algorithms, namely a protected square-and-multiply algorithm, its right-to-left counterpart, and several protected sliding-window algorithms. We also illustrate our methodology applied to point multiplication on elliptic curves. All these algorithms share the common feature that the complexity is globally unchanged compared to the corresponding unprotected implementations.

Keywords. Cryptographic algorithms, side-channel analysis, protected implementations, atomicity, exponentiation, elliptic curves.

1 Introduction

Following Goldreich [1], cryptography is concerned with the conceptualization, definition and construction of computing systems that address security concerns. We would like to add that cryptography is also concerned with concrete implementations of such systems. This in turn implies that not only the systems but also *their implementations* must withstand any abuse or misuse.

Basically, there are two main families of implementation attacks: *faults attacks* [2] and *side-channel attacks* [3, 4]. This paper only deals with the second family of attacks and more precisely with *simple* (i.e., non-differential) side-channel attacks.

⋆ Contact author.

Suppose that (part of) an algorithm consists of a loop where the execution of a given set of instructions depends on certain input values. If from some side-channel information (e.g., timing or power consumption) one can distinguish which set of instructions is processed, then one can retrieve some secret data (if any) involved during the course of the algorithm. This is the basic idea behind simple side-channel attacks.

For example, imagine that at a given step, a secret bit is used to select process Π_0 or Π_1. A straightforward counter-measure against simple side-channel attacks consists in making processes Π_0 and Π_1 indistinguishable. This is usually achieved by executing process Π_0 followed by a fake execution of process Π_1 when process Π_0 must be executed, and by executing a fake execution of process Π_0 followed by process Π_1 when process Π_1 must be executed. Such a solution is however unsatisfactory from a computational perspective because the running time can be increased by a non-negligible factor.

In a sense, our approach refines as much as possible this obvious solution. By potentially inserting dummy (fake) operations, we divide each process so that it can be expressed as the repetition of instruction blocks which appear equivalent by side-channel analysis. Such a block is called a *side-channel atomic block*. Building on this, we develop several approaches for unrolling the *whole* code so that it appears as an *uninterrupted* succession of the processes. In other words, the whole code appears as a succession of blocks that are *indistinguishable* by simple side-channel analysis. Remarkably, contrary to previous solutions, the techniques we propose are *inexpensive* and present the additional advantage of being fully *generic*, i.e., they apply to a large variety of cryptographic algorithms.

The rest of this paper is organized as follows. In the next section, we define the notion of *side-channel atomicity*. We explain how to efficiently convert a cryptographic algorithm into an algorithm protected against simple side-channel attacks. Then, in Sections 3 and 4, we provide concrete applications to the RSA cryptosystem and to elliptic curve cryptosystems. Finally, we conclude in Section 5.

2 Side-Channel Atomicity

2.1 Side-channel atomic blocks

We view a process as a sequence of instructions. We say that two instructions (or a sequence thereof) are *side-channel equivalent* if they are indistinguishable through side-channel analysis. This relation is denoted by symbol "$\sim$". From this, we define what we call a *common side-channel atomic block*.

Definition 1 (Side-channel atomicity [5]). *Given a set of processes* $\{\Pi_0, \ldots, \Pi_n\}$, *a* common side-channel atomic block Γ *for* $\Pi_0, \ldots, \Pi_n$ *is a side-channel equivalent sequence of instructions so that each process* Π_j $(0 \leq j \leq n)$ *can be expressed as the repetition of this block* Γ, *i.e., there exist sequences* $\gamma_{j,i} \sim \Gamma$ *s.t.* $\Pi_j = \gamma_{j,1} \| \gamma_{j,2} \| \cdots \| \gamma_{j,\ell_j}$. *The instruction sequences* $\gamma_{j,i}$ *are called* side-channel atomic blocks.

A common side-channel atomic block Γ always exists by noticing that one can artificially add fake instructions to an existing process to make the different processes indistinguishable. The main difficulty resides in finding a block Γ which is small with respect to some metric (e.g., running time, code size, ...). We note that a possible rearranging and/or rewriting of the processes may shorten Γ or limit the number of dummy operations and thus improve the overall performances.

2.2 Illustration

Before going further, we quote a simple example: the square-and-multiply algorithm. On input of an element x in a (multiplicatively written) group $\mathbb{G}$ and the binary expansion of exponent d, $d = (d_{m-1}, \ldots, d_0)_2$, the square-and-multiply algorithm returns $y = x^d$.

```
Input:  x, d = (d_{m-1}, ..., d_0)_2
Output: y = x^d
-------------------------------------
  R_0 ← 1 ; R_1 ← x ; i ← m − 1
  while (i ≥ 0) do
    R_0 ← (R_0)^2
    if (d_i = 1) then R_0 ← R_0 · R_1
    i ← i − 1
  endwhile
return R_0
```

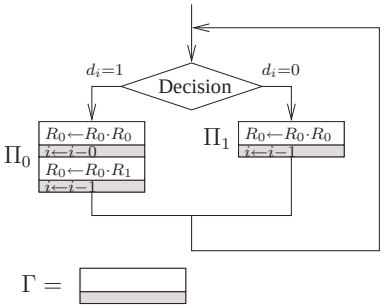
Fig. 1. (Unprotected) square-and-multiply algorithm.

As aforementioned, a valid choice for common side-channel atomic Γ consists in a squaring followed by a (possibly fake) multiplication and a counter decrementation. This algorithm is the well-known square-and-multiply *always* algorithm [6]. This is the classical way for preventing simple side-channel attacks in the square-and-multiply algorithm.

Such a choice for Γ is suboptimal. Assuming that (i) a squaring operation can be performed by calling the (hardware) multiplication routine, and (ii) instructions $R_0 \leftarrow R_0 \cdot R_0$ and $R_0 \leftarrow R_0 \cdot R_1$ are side-channel equivalent,[1] we can rewrite the previous algorithm to clearly reveal a shorter common side-channel block Γ (see Fig. 2-a). Remark the fake instruction $i \leftarrow i - 0$. Of course, we assume that this instruction is side-channel equivalent to $i \leftarrow i - 1$.

As depicted in Fig. 2-a, the algorithm is not balanced. There are two copies of Γ when $d_i = 1$ and only one when $d_i = 0$. However, as explained in the next section, it is easy to unroll the code so that a side-channel analysis only reveals a regular succession of copies of Γ without enabling to make the distinction

[1] These assumptions are discussed in Section 3.1.



(a) Synopsis.

Input: $x, d = (d_{m-1}, \ldots, d_0)_2$
Output: $y = x^d$

$R_0 \leftarrow 1$; $R_1 \leftarrow x$; $i \leftarrow m-1$
$k \leftarrow 0$
while $(i \geq 0)$ **do**
  $R_0 \leftarrow R_0 \cdot R_k$
  $k \leftarrow k \oplus d_i$; $i \leftarrow i - \neg k$
endwhile
return R_0

(b) Side-channel atomic square-and-multiply algorithm.

Fig. 2. Protected square-and-multiply algorithm.

amongst the processes being executed (i.e., Π_0 or Π_1). After simplification, we obtain the algorithm presented in Fig. 2-b.

It is worth noting that our protected algorithm (Fig. 2-b) only requires $1.5m$ multiplications, on average, for computing $y = x^d$, that is, the complexity of the usual, unprotected square-and-multiply algorithm (Fig. 1).[2]

2.3 General methodology

Given different processes $\Pi_0, \ldots, \Pi_n$, we first identify a common side-channel atomic block Γ. Next, we write the processes Π_j $(0 \leq j \leq n)$ as a repetition of Γ, i.e., $\Pi_j = \gamma_{c_j} \| \cdots \| \gamma_{c_j+\ell_j-1}$ where ℓ_j is the number of copies of Γ in Π_j,

$$\begin{cases} c_0 = 0 \\ c_j = c_{j-1} + \ell_{j-1} \quad \text{for } 1 \leq j \leq n \end{cases}$$

and with $\gamma_k \sim \Gamma$ for all $c_0 \leq k \leq c_n + \ell_n - 1$. Our strategy is to execute *exactly* ℓ_j times a sequence side-channel equivalent to Γ for process Π_j. As a result, denoting by t the running time for Γ, the time required for processing Π_j will only be $\ell_j \cdot t$ instead of $(\max_{0 \leq j \leq n} \ell_j) \cdot t$ for the trivial solution.

In order to chain the different processes, we use a bit, say s, to keep track when there are no more blocks $\gamma_k \sim \Gamma$ to be executed when processing Π_j. When process Π_j is terminated (and thus $s = 1$), we have to execute the next process according to the input values of the algorithm. Moreover, at the beginning of each loop, we update k, the number of the current sequence γ, as

$$\boxed{k \leftarrow (\neg s) \cdot (k+1) + s \cdot f(\text{input values})}$$

[2] As a side-effect, it also leaks the Hamming weight of the exponent. While this is generally not an issue, we note that the Hamming weight can be masked using standard techniques (e.g., blinding or splitting).

so that $f(\text{input values}) = c_{j'}$ if the next process to be executed is $\Pi_{j'}$. We see that when $s = 0$ then the value of k is incremented by 1. Of course, the above expression for k must be coded in such a way that no information about the input values is revealed from a given side-channel.

Alternatively, k can be defined as a counter in the current process; the updating step then becomes: $k \leftarrow (\neg s) \cdot (k+1)$. The input values are used to make the distinction amongst the different atomic blocks.

The last step consists in expressing each atomic block γ_k:

- explicitly as the elements of a table, or
- implicitly as a function of k and s (and the input values).

3 Side-Channel Atomic RSA Exponentiation

The most widely used public-key cryptosystem is the RSA [7]. Its basic operation is the (modular) exponentiation, which is usually carried out with the square-and-multiply algorithm. A side-channel atomic version of the square-and-multiply algorithm is given in Fig. 2 (see also [8]). This section presents a protected version of the ω-bit sliding-window exponentiation algorithm for any $\omega > 1$.[3] It also presents a simplified version for $\omega = 2$ as well as a right-to-left variant for $\omega = 1$. All these new algorithms use the implicit approach.

3.1 Assumptions

Our methodology supposes that a simple side-channel analysis does not allow to distinguish amongst the different atomic blocks (cf. Definition 1). As a consequence, the atomic blocks are device-dependent. From most present-day smart cards equipped with an arithmetic co-processor, our experience shows that the following operations are side-channel equivalent:[4]

1. The (modular) multiplication of two large registers: $R_i \cdot R_j$, for all i, j. This includes the case $i = j$, provided that the squaring operation is carried out by a call to the hardware multiplication (not to the hardware squaring);
2. The (modular) addition/subtraction of two large registers: $R_i \pm R_j$, for all i, j;
3. The CPU operations, i.e., all arithmetical and logical operations manipulating the CPU registers. If the hardware does not satisfy the assumption, resistance against side-channel analysis can be obtained by a software implementation. For example, if the hardware evaluation of $b \cdot A$ behaves differently according to bit $b = 0$ or 1, a simple trick consists in evaluating bA as $(b+t)A - tA$ for a random t; similarly, the addition $A \pm b$ can be evaluated as $A \pm (b+t) \mp t$;

[3] The square-and-multiply algorithm corresponds to the case $\omega = 1$.
[4] This is even more true when hardware countermeasures are activated.

4. The equality testing of two CPU registers: $A \stackrel{?}{=} B$. Again, if this is not satisfied by the hardware, a software emulation could for example read the zero flag resulting from $A \oplus B$ or perform an OR on all bits of $A \oplus B$;
5. The loading/storing of values from different registers.

[The algorithms presented in this section and in Section 4 assume hardware implementations (or software emulations thereof) satisfying the above conditions.]

3.2 Generic sliding-window algorithm

When additional registers are available, the expected amount of multiplications for evaluating $y = x^d$ can be lowered by precomputing and storing the values of x^{2j+1} for $j \in \{1, \ldots, 2^{\omega-1} - 1\}$ and then by left-to-right scanning exponent bits with an ω-bit sliding window [9, Algorithm 14.85]. This is an efficient extension of the square-and-multiply algorithm [10, 11].

Input:	$x, d = (d_{m-1}, \ldots, d_0)_2$, and an integer $\omega > 1$
Output:	$y = x^d$
Precomputation:	$R_{j+1} \leftarrow x^{2j+1}$ for $1 \leq j \leq 2^{\omega-1} - 1$

$R_0 \leftarrow 1$; $R_1 \leftarrow x$; $i \leftarrow m - 1$
for $j = 1$ **to** $\omega - 1$ **do** $d_{-j} \leftarrow 0$
$s \leftarrow 1$
while $(i \geq 0)$ **do**
  $k \leftarrow (\neg s) \cdot (k + 1)$
  $b \leftarrow 0$; $t \leftarrow 1$; $l \leftarrow \omega$; $u \leftarrow 0$
  for $j = 1$ **to** ω **do**
    $b \leftarrow b \vee d_{i-\omega+j}$; $l \leftarrow l - \neg b$
    $u \leftarrow u + t \cdot d_{i-\omega+j}$; $t \leftarrow b \cdot (2t) + \neg b$
  endfor
  $l \leftarrow l \cdot d_i$; $u \leftarrow [(u + 1) \text{ div } 2] \cdot d_i$
  $s \leftarrow (k = l)$
  $R_0 \leftarrow R_0 \cdot R_{u \cdot s}$
  $i \leftarrow i - k \cdot s - \neg d_i$
endwhile
return R_0

Fig. 3. Side-channel atomic ω-bit sliding-window algorithm.

3.3 Simplified algorithms

A larger value for ω in the ω-bit sliding-widow algorithm speeds up the computations but increases the memory requirements. A choice of particular interest for constrained devices is the case $\omega = 2$. The resulting algorithm is usually

Input: $x, d = (d_{m-1}, \ldots, d_0)_2$
Output: $y = x^d$

$R_0 \leftarrow 1$; $R_1 \leftarrow x$; $R_2 \leftarrow x^3$
$d_{-1} \leftarrow 0$; $i \leftarrow m-1$; $s \leftarrow 1$
while $(i \geq 0)$ **do**
  $k \leftarrow (\neg s) \cdot (k+1)$
  $s \leftarrow s \oplus d_i \oplus (d_{i-1} \wedge (k \bmod 2))$
  $R_0 \leftarrow R_0 \cdot R_{k \cdot s}$
  $i \leftarrow i - k \cdot s - \neg d_i$
endwhile
return R_0

(a) Side-channel atomic (M, M^3) algorithm.

Input: $x, d = (d_{m-1}, \ldots, d_0)_2$
Output: $y = x^d$

$R_0 \leftarrow 1$; $R_1 \leftarrow x$; $i \leftarrow 0$
$k \leftarrow 1$
while $(i \leq m-1)$ **do**
  $k \leftarrow k \oplus d_i$
  $R_k \leftarrow R_k \cdot R_1$
  $i \leftarrow i + k$
endwhile
return R_0

(b) Side-channel atomic right-to-left binary algorithm.

Fig. 4. Further simplified algorithms for constrained devices.

referred to as the (M, M^3) algorithm. The generic algorithm of Fig. 3 can then be simplified to the algorithm given in Fig. 4-a.

In some cases, it is easier to scan bits from the least significant position to the most significant one. There is a right-to-left analogue of the square-and-multiply algorithm for computing $y = x^d$. Analogously to Fig. 2, we can modify it into an algorithm preventing simple side-channel attacks. After simplification, we get the protected right-to-left exponentiation algorithm given in Fig. 4-b.

There are of course numerous possible variants which may be more efficient on a particular given architecture. What is remarkable is that our protected algorithms have roughly the *same* complexity (running time and memory requirements) as their respective unprotected versions.

4 Side Channel Atomic Elliptic Curve Point Multiplication

Our methodology applies to virtually any algorithm. We show hereafter how to adapt it in the context of elliptic curve cryptography [12]. Two categories of elliptic curves are commonly used [13]: elliptic curves over large prime fields and non-supersingular elliptic curves over binary fields. The basic operation in elliptic curve cryptography consists in computing the multiple of a point, that is, given a point $\boldsymbol{P_1}$ on an elliptic curve, one has to compute $\boldsymbol{P_d} = d\boldsymbol{P_1}$. To ease the presentation, we assume that this is carried out with the (additive version of the) square-and-multiply algorithm. Other methods are discussed in [14]. Our methodology readily applies to those implementation choices as well.

4.1 Elliptic curves defined over large prime fields

Consider the elliptic curve E defined over a prime field $\mathbb{F}_p$ (with $p > 3$) given by the Weierstraß equation

$$E_{/\mathbb{F}_p} : y^2 = x^3 + ax + b \ .$$

To avoid field inversion, Jacobian coordinates are generally used [13] for representing points on E. With Jacobian coordinates, the doubling of $\boldsymbol{P_1}$ is $2(X_1, Y_1, Z_1) = (X_3, Y_3, Z_3)$ where

$$X_3 = M^2 - 2S,\ Y_3 = M(S - X_3) - T,\ Z_3 = 2Y_1Z_1$$

with $M = 3X_1^2 + aZ_1^4$, $S = 4X_1Y_1^2$ and $T = 8Y_1^4$. The sum of two (distinct) points $\boldsymbol{P_1} = (X_1, Y_1, Z_1)$ and $\boldsymbol{P_2} = (X_2, Y_2, Z_2)$ is (X_3, Y_3, Z_3) where

$$\begin{gathered} X_3 = W^3 - 2U_1W^2 + R^2, \\ Y_3 = -S_1W^3 + R(U_1W^2 - X_3), Z_3 = Z_1Z_2W \end{gathered}$$

with $U_1 = X_1Z_2^2$, $U_2 = X_2Z_1^2$, $S_1 = Y_1Z_2^3$, $S_2 = Y_2Z_1^3$, $W = U_1 - U_2$ and $R = S_1 - S_2$.

As the operations doubling or adding points are somewhat involved, we adopt the explicit approach. We refer the reader to the appendix for the detailed formulæ leading to the expression of atomic blocks γ_k as the rows of matrix:

$$(u^*_{k,l})_{\substack{0\le k\le 25\\ 0\le l\le 9}} = \begin{pmatrix}
4&1&1&5&4&4&3&4&4&5\\
5&3&3&1&1&1&3&1&1&3\\
5&5&5&1&1&3&3&1&1&3\\
5&0&5&4&4&5&3&5&2&2\\
3&3&5&1&1&3&3&1&1&3\\
2&2&2&2&2&2&4&1&1&3\\
5&1&2&1&1&5&5&1&1&5\\
1&4&4&1&1&5&4&1&1&5\\
2&2&2&2&2&2&3&5&1&5\\
4&4&5&2&2&4&2&4&4&5\\
4&9&9&5&1&5&5&5&1&5\\
1&1&4&5&1&5&5&5&1&5\\
4&4&9&5&1&5&5&5&1&5\\
2&2&4&5&1&5&5&5&1&5\\
4&3&3&5&1&5&5&5&1&5\\
5&4&7&2&2&5&5&5&1&5\\
4&3&4&2&2&5&6&6&5&6\\
4&4&8&6&5&6&4&4&2&4\\
3&3&9&6&5&6&6&6&5&6\\
3&3&5&6&5&6&6&6&5&6\\
6&5&5&6&3&6&3&6&3&6\\
1&1&6&1&1&4&4&1&1&4\\
5&5&6&6&1&2&2&6&2&6\\
1&4&4&1&1&5&6&1&1&6\\
2&2&5&1&1&6&3&6&1&6\\
4&4&6&2&2&4&6&6&1&6
\end{pmatrix}.$$

The resulting algorithm is given in the next figure.

Input: $\boldsymbol{P_1} = (X_1, Y_1, Z_1), d = (1, d_{m-2}, \ldots, d_0)_2$, and matrix $(u^*_{k,l})$ as above
Output: $\boldsymbol{P_d} = d\boldsymbol{P_1}$

$R_0 \leftarrow a$; $R_1 \leftarrow X_1$; $R_2 \leftarrow Y_1$; $R_3 \leftarrow Z_1$; $R_7 \leftarrow X_1$; $R_8 \leftarrow Y_1$; $R_9 \leftarrow Z_1$
$i \leftarrow m-2$; $s \leftarrow 1$
while $(i \geq 0)$ **do**
  $k \leftarrow (\neg s) \cdot (k+1)$
  $s \leftarrow d_i \cdot (k \operatorname{div} 25) + (\neg d_i) \cdot (k \operatorname{div} 9)$
  $R_{u^*_{k,0}} \leftarrow R_{u^*_{k,1}} \cdot R_{u^*_{k,2}}$; $R_{u^*_{k,3}} \leftarrow R_{u^*_{k,4}} + R_{u^*_{k,5}}$; $R_{u^*_{k,6}} \leftarrow -R_{u^*_{k,6}}$; $R_{u^*_{k,7}} \leftarrow R_{u^*_{k,8}} + R_{u^*_{k,9}}$
  $i \leftarrow i - s$
endwhile
return (R_1, R_2, R_3)

Fig. 5. Side-channel atomic double-and-add algorithm for elliptic curves over $\mathbb{F}_p$.

Again, it is worth noting that, in terms of field multiplications, this algorithm is as efficient as the corresponding unprotected implementation (cf. [13]).

4.2 Elliptic curves defined over a binary field

Atomicity is a relative notion. In the previous examples, we considered addition and multiplication as basic operations. One could also imagine that division is an basic operation; for instance, if division is provided by a dedicated hardware routine. The more different (from a side-channel perspective) basic operations there are, the more difficult it is to exhibit a common side-channel atomic block Γ. We give hereafter an example involving a division (a rather costly operation) as basic operation.

For efficiency reasons, it is recommended to use affine coordinates for adding points on an elliptic curve defined over a binary field [15]. A (non-supersingular) elliptic curve E defined over the binary field $\mathbb{F}_{2^q}$ is given by the Weierstraß equation

$$E_{/\mathbb{F}_{2^q}} : y^2 + xy = x^3 + ax^2 + b \ .$$

In affine coordinates (cf. [13]), the doubling of point $\boldsymbol{P_1} = (x_1, y_1)$ is given by $2(x_1, y_1) = (x_3, y_3)$ where

$$x_3 = a + \lambda^2 + \lambda, \quad y_3 = (x_1 + x_3)\lambda + x_3 + y_1$$

with $\lambda = x_1 + (y_1/x_1)$. The sum of two (distinct) points $\boldsymbol{P_1} = (x_1, y_1)$ and $\boldsymbol{P_2} = (x_2, y_2)$ is (x_3, y_3) where

$$x_3 = a + \lambda^2 + \lambda + x_1 + x_2, \quad y_3 = (x_1 + x_3)\lambda + x_3 + y_1$$

with $\lambda = (y_1 + y_2)/(x_1 + x_2)$. We clearly see that doubling and addition of points can be made very similar. As a result, we choose for Γ a whole elliptic curve

doubling or addition (see Fig. 6-a). Only two extra (field) additions are needed for doubling a point compared to the unprotected version of [13].

From this, an efficient protected double-and-add algorithm can then be derived (see Fig. 6-b).

Input: $(T_1, T_2) = \boldsymbol{P_1}$, $(T_3, T_4) = \boldsymbol{P_2}$
Output: $\boldsymbol{P_1} + \boldsymbol{P_2}$ or $2\boldsymbol{P_1}$

Addition: $\boldsymbol{P_1} \leftarrow \boldsymbol{P_2} + \boldsymbol{P_1}$	Doubling: $\boldsymbol{P_1} \leftarrow 2\boldsymbol{P_1}$
$T_1 \leftarrow T_1 + T_3$ $(= x_1 + x_2)$	$T_6 \leftarrow T_1 + T_3$ (fake)
$T_2 \leftarrow T_2 + T_4$ $(= y_1 + y_2)$	$T_6 \leftarrow T_3 + T_6$ $(= x_1)$
$T_5 \leftarrow T_2/T_1$ $(= \lambda)$	$T_5 \leftarrow T_2/T_1$ $(= y_1/x_1)$
$T_1 \leftarrow T_1 + T_5$	$T_5 \leftarrow T_1 + T_5$ $(= \lambda)$
$T_6 \leftarrow {T_5}^2$ $(= \lambda^2)$	$T_1 \leftarrow {T_5}^2$ $(= \lambda^2)$
$T_6 \leftarrow T_6 + a$ $(= \lambda^2 + a)$	$T_1 \leftarrow T_1 + a$ $(= \lambda^2 + a)$
$T_1 \leftarrow T_1 + T_6$ $(= x_3)$	$T_1 \leftarrow T_1 + T_5$ $(= x_3)$
$T_2 \leftarrow T_1 + T_4$ $(= x_3 + y_2)$	$T_2 \leftarrow T_1 + T_2$ $(= x_3 + y_1)$
$T_6 \leftarrow T_1 + T_3$ $(= x_2 + x_3)$	$T_6 \leftarrow T_1 + T_6$ $(= x_1 + x_3)$
$T_5 \leftarrow T_5 \cdot T_6$	$T_5 \leftarrow T_5 \cdot T_6$
$T_2 \leftarrow T_2 + T_5$ $(= y_3)$	$T_2 \leftarrow T_2 + T_5$ $(= y_3)$
return (T_1, T_2)	

(a) Side-channel atomic elliptic curve addition[5] for elliptic curves over $\mathbb{F}_{2^q}$.

Input: $\boldsymbol{P_1} = (x_1, y_1), d = (1, d_{m-2}, \ldots, d_0)_2$
Output: $\boldsymbol{P_d} = d\boldsymbol{P_1}$

$R_1 \leftarrow x_1$; $R_2 \leftarrow y_1$; $R_3 \leftarrow x_1$; $R_4 \leftarrow y_1$
$i \leftarrow m - 2$; $s \leftarrow 1$
while $(i \geq 0)$ **do**
  $k \leftarrow (\neg s) \cdot (k + 1)$; $s \leftarrow k \vee (\neg d_i)$
  $R_{6-5k} \leftarrow R_1 + R_3$; $R_{6-4k} \leftarrow R_{3-k} + R_{6-2k}$
  $R_5 \leftarrow R_2/R_1$
  $R_{5-4k} \leftarrow R_1 + R_5$
  $R_{1+5k} \leftarrow (R_5)^2$
  $R_{1+5k} \leftarrow R_{1+5k} + a$
  $R_1 \leftarrow R_1 + R_{5+k}$
  $R_2 \leftarrow R_1 + R_{2+2k}$; $R_6 \leftarrow R_1 + R_{6-3k}$
  $R_5 \leftarrow R_5 \cdot R_6$; $R_2 \leftarrow R_2 + R_5$
  $i \leftarrow i - s$
endwhile
return (R_1, R_2)

(b) Side-channel atomic double-and-add algorithm for elliptic curves over $\mathbb{F}_{2^q}$.

Fig. 6. Side-channel atomic elliptic curve algorithms.

5 Conclusion

This paper introduced the notion of common side-channel atomicity. Based on this, novel solutions towards resistance against side-channel attacks were presented. The proposed solutions are generic and apply to a large variety of cryptographic systems. In particular, they apply to exponentiation-based systems for which they lead to protected algorithms having roughly the same efficiency as their straightforward (i.e., unprotected) implementations. Finally, it should be noted that our methodology nicely combines with countermeasures against the more sophisticated differential analysis.

[5] In order to save a register, we take advantage of commutativity by computing $\boldsymbol{P_1} \leftarrow \boldsymbol{P_2} + \boldsymbol{P_1}$ instead of $\boldsymbol{P_1} \leftarrow \boldsymbol{P_1} + \boldsymbol{P_2}$ for the elliptic curve addition.

Acknowledgements

We would like to thank the anonymous referees for their useful comments. Part of this work was performed while the second author was visiting Gemplus. Thanks go to David Naccache, Philippe Proust and Jean-Jacques Quisquater for making this arrangement possible.

References

1. O. Goldreich. *Foundations of Cryptography – Basic Tools*. Cambridge University Press, 2001.
2. D. Boneh, R.A. DeMillo, and R.J. Lipton. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology – EUROCRYPT '97*, vol. 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.
3. P.C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO '96*, vol. 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.
4. P.C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology – CRYPTO '99*, vol. 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
5. B. Chevallier-Mames and M. Joye. Procédé cryptographique protégé contre les attaques de type à canal caché. Demande de brevet français, no. 0204117, April 2002.
6. J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems (CHES '99)*, vol. 1717 of *Lecture Note in Computer Science*, pages 292–302. Springer-Verlag, 1999.
7. R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* **21**(2):120–126, 1976.
8. M. Joye. Recovering lost efficiency of exponentiation algorithms on smart cards. *Electronics Letters* **38**(19):1095–1097, 2002.
9. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
10. L.-C.-K. Hui and K.-Y. Lam. Fast square-and-multiply exponentiation for RSA. *Electronics Letters* **30**(17):1396–1397, 1994.
11. K.-Y. Lam and L.-C.-K. Hui. Efficiency of $SS(l)$ square-and-multiply exponentiation algorithms. *Electronics Letters* **30**(25):2115–2116, 1994.
12. I. Blake, G. Seroussi, and N.P. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.
13. IEEE Std 1363-2000. *IEEE Standard Specifications for Public-Key Cryptography*. IEEE Computer Society, August 29, 2000.
14. D.M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms* **27**:129–146, 1998.
15. E. De Win, S. Mister, B. Preneel, and M. Wiener. *On the performance of signature schemes based on elliptic curves*. In *Algorithmic Number Theory Symposium*, vol. 1423 of *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 1998.
16. M. Joye and C. Tymen. Protections against differential analysis for elliptic curve cryptography: An algebraic approach. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, vol. 2162 of *Lecture Notes in Computer Science*, pages 377–390. Springer-Verlag, 2001.

A Appendix

This appendix details how matrix $(u^*_{k,l})$ used in the double-and-add algorithm of Fig. 5 was obtained.

From the point addition formulæ given in Section 4.1, we see that doubling a point requires 10 multiplications and adding two points requires 16 multiplications. In addition to multiplications, adding or doubling points also involve (field) additions/subtractions. Consequently, a common side-channel atomic block, Γ, must at least include one multiplication and one addition (a subtraction can be considered as a special case of negation followed by an addition). Since 1) the formula for adding two (distinct) points requires more multiplications than (field) additions, and 2) the formula for doubling a point requires 11 (field) addition/subtraction, we choose to express Γ with 1 (field) multiplication and 2 (field) additions (along with a negation to possibly perform a subtraction).

We now express the point doubling and point addition as a repetition of blocks side-channel equivalent to Γ (see Fig. 7). A '$\star$' indicates that any register that does not disturb the course of the algorithm can be selected.

Replacing the '$\star$' by appropriate choices, process Π_0 (doubling followed by an addition) and process Π_1 (doubling) in the double-and-add algorithm can be defined by matrix

$$(u_{k,l})_{\substack{0\le k\le 35\\ 0\le l\le 10}} = \begin{pmatrix}
4&1&1&5&4&4&3&4&4&5&0\\
5&3&3&1&1&1&3&1&1&3&0\\
5&5&5&1&1&3&3&1&1&3&0\\
5&0&5&4&4&5&3&5&2&2&0\\
3&3&5&1&1&3&3&1&1&3&0\\
2&2&2&2&2&2&4&1&1&3&0\\
5&1&2&1&1&5&5&1&1&5&0\\
1&4&4&1&1&5&4&1&1&5&0\\
2&2&2&2&2&2&3&5&1&5&0\\
4&4&5&2&2&4&2&4&4&5&0\\
4&9&9&5&1&5&5&5&1&5&0\\
1&1&4&5&1&5&5&5&1&5&0\\
4&4&9&5&1&5&5&5&1&5&0\\
2&2&4&5&1&5&5&5&1&5&0\\
4&3&3&5&1&5&5&5&1&5&0\\
5&4&7&2&2&5&5&5&1&5&0\\
4&3&4&2&2&5&6&6&5&6&0\\
4&4&8&6&5&6&4&4&2&4&0\\
3&3&9&6&5&6&6&6&5&6&0\\
3&3&5&6&5&6&6&6&5&6&0\\
6&5&5&6&3&6&3&6&3&6&0\\
1&1&6&1&1&4&4&1&1&4&0\\
5&5&6&6&1&2&2&6&2&6&0\\
1&4&4&1&1&5&6&1&1&6&0\\
2&2&5&1&1&6&3&6&1&6&0\\
4&4&6&2&2&4&6&6&1&6&1\\
\hline
4&1&1&5&4&4&3&4&4&5&0\\
5&3&3&1&1&1&3&1&1&3&0\\
5&5&5&1&1&3&3&1&1&3&0\\
5&0&5&4&4&5&3&5&2&2&0\\
3&3&5&1&1&3&3&1&1&3&0\\
2&2&2&2&2&2&4&1&1&3&0\\
5&1&2&1&1&5&5&1&1&5&0\\
1&4&4&1&1&5&4&1&1&5&0\\
2&2&2&2&2&2&3&5&1&5&0\\
4&4&5&2&2&4&2&4&4&5&1
\end{pmatrix}$$

Point doubling

$T_0 \leftarrow a,\ T_1 \leftarrow X_1,\ T_2 \leftarrow Y_1,\ T_3 \leftarrow Z_1$

1. $\begin{bmatrix} T_4 \leftarrow T_1 \cdot T_1 & (= X_1^2) \\ T_5 \leftarrow T_4 + T_4 & (= 2X_1^2) \\ \star \\ T_4 \leftarrow T_4 + T_5 & (= 3X_1^2) \end{bmatrix}$
2. $\begin{bmatrix} T_5 \leftarrow T_3 \cdot T_3 & (= Z_1^2) \\ T_1 \leftarrow T_1 + T_1 & (= 2X_1) \\ \star \\ \star \end{bmatrix}$
3. $\begin{bmatrix} T_5 \leftarrow T_5 \cdot T_5 & (= Z_1^4) \\ \star \\ \star \\ \star \end{bmatrix}$
4. $\begin{bmatrix} T_5 \leftarrow T_0 \cdot T_5 & (= aZ_1^4) \\ T_4 \leftarrow T_4 + T_5 & (= M) \\ \star \\ T_5 \leftarrow T_2 + T_2 & (= 2Y_1) \end{bmatrix}$
5. $\begin{bmatrix} T_3 \leftarrow T_3 \cdot T_5 & (= Z_2) \\ \star \\ \star \\ \star \end{bmatrix}$
6. $\begin{bmatrix} T_2 \leftarrow T_2 \cdot T_2 & (= Y_1^2) \\ T_2 \leftarrow T_2 + T_2 & (= 2Y_1^2) \\ \star \\ \star \end{bmatrix}$
7. $\begin{bmatrix} T_5 \leftarrow T_1 \cdot T_2 & (= S) \\ \star \\ T_5 \leftarrow -T_5 & (= -S) \\ \star \end{bmatrix}$
8. $\begin{bmatrix} T_1 \leftarrow T_4 \cdot T_4 & (= M^2) \\ T_1 \leftarrow T_1 + T_5 & (= M^2 - S) \\ \star \\ T_1 \leftarrow T_1 + T_5 & (= X_2) \end{bmatrix}$
9. $\begin{bmatrix} T_2 \leftarrow T_2 \cdot T_2 & (= 4Y_1^4) \\ T_2 \leftarrow T_2 + T_2 & (= T) \\ \star \\ T_5 \leftarrow T_1 + T_5 & (= X_2 - S) \end{bmatrix}$
10. $\begin{bmatrix} T_4 \leftarrow T_4 \cdot T_5 & (= -Y_2 - T) \\ T_2 \leftarrow T_2 + T_4 & (= -Y_2) \\ T_2 \leftarrow -T_2 & (= Y_2) \\ \star \end{bmatrix}$

These formulæ assume that multiplication by parameter a (cf. Step 4) behaves identically as a multiplication with another value. If this multiplication can be distinguished by side-channel analysis, elliptic curve operations can be performed on a randomly chosen isomorphic curve [16]. Provided that multiplication by $-a$ cannot be distinguished, another way to prevent side-channel leakage is to replace above Steps 2, 3 and 4 by

2. $\begin{bmatrix} T_5 \leftarrow T_3 \cdot T_3 \\ T_1 \leftarrow T_1 + T_1 \\ T_0 \leftarrow -T_0 \\ \star \end{bmatrix}$ 3. $\begin{bmatrix} T_5 \leftarrow T_5 \cdot T_5 \\ \star \\ T_5 \leftarrow -T_5 \\ \star \end{bmatrix}$ 4. $\begin{bmatrix} T_5 \leftarrow T_0 \cdot T_5 \\ T_4 \leftarrow T_4 + T_5 \\ T_0 \leftarrow -T_0 \\ T_5 \leftarrow T_2 + T_2 \end{bmatrix}$.

Point addition

$T_1 \leftarrow X_1,\ T_2 \leftarrow Y_1,\ T_3 \leftarrow Z_1,\ T_7 \leftarrow X_2,\ T_8 \leftarrow Y_2,\ T_9 \leftarrow Z_2$

1. $\begin{bmatrix} T_4 \leftarrow T_9 \cdot T_9 & (= Z_2^2) \\ \star \\ \star \\ \star \end{bmatrix}$
2. $\begin{bmatrix} T_1 \leftarrow T_1 \cdot T_4 & (= U_1) \\ \star \\ \star \\ \star \end{bmatrix}$
3. $\begin{bmatrix} T_4 \leftarrow T_4 \cdot T_9 & (= Z_2^3) \\ \star \\ \star \\ \star \end{bmatrix}$
4. $\begin{bmatrix} T_2 \leftarrow T_2 \cdot T_4 & (= S_1) \\ \star \\ \star \\ \star \end{bmatrix}$
5. $\begin{bmatrix} T_4 \leftarrow T_3 \cdot T_3 & (= Z_1^2) \\ \star \\ \star \\ \star \end{bmatrix}$
6. $\begin{bmatrix} T_5 \leftarrow T_4 \cdot T_7 & (= U_2) \\ \star \\ T_5 \leftarrow -T_5 & (= -U_2) \\ T_5 \leftarrow T_1 + T_5 & (= W) \end{bmatrix}$
7. $\begin{bmatrix} T_4 \leftarrow T_3 \cdot T_4 & (= Z_1^3) \\ \star \\ \star \\ \star \end{bmatrix}$
8. $\begin{bmatrix} T_4 \leftarrow T_4 \cdot T_8 & (= S_2) \\ \star \\ T_4 \leftarrow -T_4 & (-S_2) \\ T_4 \leftarrow T_2 + T_4 & (= R) \end{bmatrix}$
9. $\begin{bmatrix} T_3 \leftarrow T_3 \cdot T_9 & (= Z_1 Z_2) \\ \star \\ \star \\ \star \end{bmatrix}$
10. $\begin{bmatrix} T_3 \leftarrow T_3 \cdot T_5 & (= Z_3) \\ \star \\ \star \\ \star \end{bmatrix}$
11. $\begin{bmatrix} T_6 \leftarrow T_5 \cdot T_5 & (= W^2) \\ \star \\ \star \\ \star \end{bmatrix}$
12. $\begin{bmatrix} T_1 \leftarrow T_1 \cdot T_6 & (= U_1 W^2) \\ \star \\ T_4 \leftarrow -T_4 & (= -R) \\ \star \end{bmatrix}$
13. $\begin{bmatrix} T_5 \leftarrow T_5 \cdot T_6 & (= W^3) \\ T_6 \leftarrow T_1 + T_2 & (= S_1 + U_1 W^2) \\ T_2 \leftarrow -T_2 & (= -S_1) \\ T_6 \leftarrow T_2 + T_6 & (= U_1 W^2) \end{bmatrix}$
14. $\begin{bmatrix} T_1 \leftarrow T_4 \cdot T_4 & (= R^2) \\ T_1 \leftarrow T_1 + T_5 & (= R^2 + W^3) \\ T_6 \leftarrow -T_6 & (= -U_1 W^2) \\ T_1 \leftarrow T_1 + T_6 & (= X_3 + U_1 W^2) \end{bmatrix}$
15. $\begin{bmatrix} T_2 \leftarrow T_2 \cdot T_5 & (= -S_1 W^3) \\ T_1 \leftarrow T_1 + T_6 & (= X_3) \\ \star \\ T_6 \leftarrow T_1 + T_6 & (= X_3 - U_1 W^2) \end{bmatrix}$
16. $\begin{bmatrix} T_4 \leftarrow T_4 \cdot T_6 & (= Y_3 + S_1 W^3) \\ T_2 \leftarrow T_2 + T_4 & (= Y_3) \\ \star \\ \star \end{bmatrix}$

Fig. 7. Expressing point doubling and point addition as of repetition of blocks $\sim \Gamma$.

whose k^{th} row represents sequence γ_k, which reads as

$$\gamma_k = [R_{u_{k,0}} \leftarrow R_{u_{k,1}} \cdot R_{u_{k,2}}; R_{u_{k,3}} \leftarrow R_{u_{k,4}} + R_{u_{k,5}}; \\ R_{u_{k,6}} \leftarrow -R_{u_{k,6}}; R_{u_{k,7}} \leftarrow R_{u_{k,8}} + R_{u_{k,9}}; i \leftarrow i - u_{k,10}] \ .$$

So, a direct application yields the following implementation of the double-and-add algorithm.

Input: $\boldsymbol{P_1} = (X_1, Y_1, Z_1), d = (1, d_{m-2}, \ldots, d_0)_2$, and matrix $(u_{k,l})$ as above
Output: $\boldsymbol{P_d} = d\boldsymbol{P_1}$

$R_0 \leftarrow a$; $R_1 \leftarrow X_1$; $R_2 \leftarrow Y_1$; $R_3 \leftarrow Z_1$; $R_7 \leftarrow X_1$; $R_8 \leftarrow Y_1$; $R_9 \leftarrow Z_1$
$i \leftarrow m - 2$; $s \leftarrow 1$
while $(i \geq 0)$ do
  $k \leftarrow (\neg s) \cdot (k + 1) + s \cdot 26(\neg d_i)$
  $(u_0, u_1, \ldots, u_9, s) \leftarrow (u_{k,0}, u_{k,1}, \ldots, u_{k,9}, u_{k,10})$
  $R_{u_0} \leftarrow R_{u_1} \cdot R_{u_2}$; $R_{u_3} \leftarrow R_{u_4} + R_{u_5}$; $R_{u_6} \leftarrow -R_{u_6}$; $R_{u_7} \leftarrow R_{u_8} + R_{u_9}$
  $i \leftarrow i - s$
return (R_1, R_2, R_3)

Fig. 8. A [simple] side-channel atomic double-and-add algorithm for elliptic curves over $\mathbb{F}_p$.

Matrix $(u_{k,l})$ is highly redundant: except for variable s (last column), the first 10 rows are exactly the same as the last 10 rows. This is not too surprising since these rows correspond to the same operation (namely, an elliptic curve doubling). It is fairly easy to remove the redundancy. Since, except for s, the 10 rows representing a doubling in matrix $(u_{k,l})$ are equivalent, they can be shared. It suffices then to express s as a function of d_i and k in the optimized matrix $(u^*_{k,l})$ given by

$$(u^*_{k,l})_{\substack{0 \leq k \leq 25 \\ 0 \leq l \leq 9}} = \begin{pmatrix} 4&1&1&5&4&4&3&4&4&5 \\ 5&3&3&1&1&1&3&1&1&3 \\ 5&5&5&1&1&3&3&1&1&3 \\ 5&0&5&4&4&5&3&5&2&2 \\ 3&3&5&1&1&3&3&1&1&3 \\ 2&2&2&2&2&2&4&1&1&3 \\ 5&1&2&1&1&5&5&1&1&5 \\ 1&4&4&1&1&5&4&1&1&5 \\ 2&2&2&2&2&2&3&5&1&5 \\ 4&4&5&2&2&4&2&4&4&5 \\ 4&9&9&5&1&5&5&5&1&5 \\ 1&1&4&5&1&5&5&5&1&5 \\ 4&4&9&5&1&5&5&5&1&5 \\ 2&2&4&5&1&5&5&5&1&5 \\ 4&3&3&5&1&5&5&5&1&5 \\ 5&4&7&2&2&5&5&5&1&5 \\ 4&3&4&2&2&5&6&6&5&6 \\ 4&4&8&6&5&6&4&4&2&4 \\ 3&3&9&6&5&6&6&6&5&6 \\ 3&3&5&6&5&6&6&6&5&6 \\ 6&5&5&6&3&6&3&6&3&6 \\ 1&1&6&1&1&4&4&1&1&4 \\ 5&5&6&6&1&2&2&6&2&6 \\ 1&4&4&1&1&5&6&1&1&6 \\ 2&2&5&1&1&6&3&6&1&6 \\ 4&4&6&2&2&4&6&6&1&6 \end{pmatrix} .$$

Since, when $d_i = 1$ we have $s = 0$ if $0 \leq k \leq 24$ and $s = 1$ if $k = 25$, and when $d_i = 0$ we have $s = 0$ if $0 \leq k \leq 8$ and $s = 1$ if $k = 9$, we may for example define s as

$$s = d_i \cdot (k \text{ div } 25) + (\neg d_i) \cdot (k \text{ div } 9) \ .$$

The expression for k must also be modified accordingly: k is always incremented unless when $s = 1$, in which case it must be set to 0. So, $k \leftarrow (\neg s) \cdot (k + 1)$ is a valid expression for updating k. Doing so, we obtain the algorithm of Fig. 5, which is very similar to the above above but with a smaller matrix representation (i.e., matrix $(u^*_{k,l})$).

Hessian Elliptic Curves and Side-Channel Attacks

[Published in Ç.K. Koç, D. Naccache, and C. Paar, Eds., *Cryptographic Hardware and Embedded Systems – CHES 2001*, vol. 2162 of *Lecture Notes in Computer Science*, pp. 402–410, Springer-Verlag, 2001.]

Marc Joye[1] and Jean-Jacques Quisquater[2]

[1] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
marc.joye@gemplus.com
http://www.geocities.com/MarcJoye/
[2] UCL Crypto Group, Université catholique de Louvain
Place du Levant 3, 1348 Louvain-la-Neuve, Belgium
jjq@dice.ucl.ac.be

Abstract. Side-channel attacks are a recent class of attacks that have been revealed to be very powerful in practice. By measuring some side-channel information (running time, power consumption, ...), an attacker is able to recover some secret data from a carelessly implemented crypto-algorithm. This paper investigates the Hessian parameterization of an elliptic curve as a step towards resistance against such attacks in the context of elliptic curve cryptography. The idea is to use the same procedure to compute the addition, the doubling or the subtraction of points. As a result, this gives a 33% performance improvement as compared to the best reported methods and requires much less memory.

Keywords. Elliptic curves, cryptography, side-channel attacks, implementation, smart cards.

1 Introduction

Side-channel attacks are a recent class of attacks that have been revealed to be very powerful in practice. By measuring some side-channel information (running time, power consumption, ...), an attacker is able to recover some secret data from a carelessly implemented crypto-algorithm. This paper investigates the Hessian parameterization of an elliptic curve as a step towards resistance against such attacks in the context of elliptic curve cryptography. The idea is to use the same procedure to compute the addition, the doubling or the subtraction of points. As a result, this gives a 33% performance improvement as compared to the best reported methods and requires much less memory.

The rest of this paper is organized as follows. The next section introduces the theory of elliptic curves and reviews the related work for computing the multiple of a point on an elliptic curve. Section 3 presents the Hessian parameterization of an elliptic curve. It also proves some useful results on this special parameterization. The side-channel attacks are defined in Section 4 and some countermeasures are discussed. Finally, Section 5 shows how the Hessian parameterization helps to efficiently foil such attacks in the context of elliptic curve cryptography.

2 Elliptic Curve Multiplication

To ease the exposition we assume throughout this paper that $\mathbb{K}$ is a field of characteristic $p > 3$.

2.1 Basic facts

We start with a short introduction to elliptic curves.

Definition 1. *Up to a birational equivalence, an* elliptic curve *over a field* $\mathbb{K}$ *is a plane nonsingular cubic curve with a* $\mathbb{K}$*-rational point.*

Elliptic curves are often expressed in terms of Weierstraß equations:

$$E_{/\mathbb{K}} : y^2 = x^3 + ax + b \quad (\text{with } 4a^3 + 27b^2 \neq 0) \tag{1}$$

where a and $b \in \mathbb{K}$. The condition $4a^3 + 27b^2 \neq 0$ ensures that the *discriminant*

$$\Delta = -16(4a^3 + 27b^2) \tag{2}$$

is nonzero, or equivalently that the points (x, y) on the curve are nonsingular.

More importantly, together with the *point at infinity* $\boldsymbol{O}$, the points of an elliptic curve form an Abelian group (with identity element $\boldsymbol{O}$) under the *chord-and-tangent rule* defined as follows. If $\boldsymbol{P} = (x_1, y_1)$, then its inverse is given by $-\boldsymbol{P} = (x_1, -y_1)$. The sum of two points $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2)$ (with $\boldsymbol{Q} \neq -\boldsymbol{P}$) is equal to $\boldsymbol{R} = (x_3, y_3)$ where

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \lambda(x_1 - x_3) - y_1$$

$$\text{with } \lambda = \begin{cases} \dfrac{3x_1^2 + a}{2y_1} & \text{if } x_1 = x_2\,, \\ \dfrac{y_1 - y_2}{x_1 - x_2} & \text{otherwise}\ . \end{cases}$$

The previous formulæ require 2 or 3 multiplications and 1 inversion to add two points. Since this latter operation is costly (an inversion roughly takes the same amount of time as 23 multiplications [8]), *projective* representations of Weierstraß equations may be preferred.

3 Hessian Curves

In this section, we formally define the Hessian elliptic curves [10] (see also [2, p. 36] and [15]) and give some results on this special parameterization.

Definition 2. *An* Hessian elliptic curve *over* $\mathbb{K}$ *is a plane cubic curve given by an equation of the form*

$$E_{/\mathbb{K}} : u^3 + v^3 + 1 = 3Duv\,, \tag{3}$$

or in projective coordinates,

$$E_{/\mathbb{K}} : U^3 + V^3 + W^3 = 3DUVW \tag{4}$$

where $D \in \mathbb{K}$ *and* $D^3 \neq 1$.

As shown in the next lemma, the condition $D^3 \neq 1$ imposes that the curve is nonsingular, that is, elliptic.

Lemma 1. *An Hessian cubic curve* $E_D(\mathbb{K})$ *is singular if and only if* $D^3 = 1$.

Proof. Let $\boldsymbol{P} = (U_1 : V_1 : W_1)$ be a singular point. Then $U_1^2 - DV_1W_1 = V_1^2 - DU_1W_1 = W_1^2 - DV_1W_1 = 0$, hence $U_1^3 = V_1^3 = W_1^3$ $(\neq 0)$. Therefore there exist $k \in \mathbb{K}^*$ and $r, s, t \in \mathbb{Z}_3$ such that $U_1 = k\omega^r$, $V_1 = k\omega^s$ and $W_1 = k\omega^t$ where ω is a non-trivial cubic root of unity. Together with Eq. (4), this yields $3k^3 = 3Dk^3\omega^{r+s+t}$, or equivalently, $D^3 = 1$. □

Proposition 1. *The Hessian curve given by Eq. (3) is birationnally equivalent to the Weierstraß equation*

$$y^2 = x^3 - 27D(D^3 + 8)x + 54(D^6 - 20D^3 - 8)\,, \tag{5}$$

under the transformations

$$(u, v) = \big(\eta(x + 9D^2), -1 + \eta(3D^3 - Dx - 12)\big) \tag{6}$$

and

$$(x, y) = \big(-9D^2 + \xi u, 3\xi(v - 1)\big) \tag{7}$$

where $\eta = \frac{6(D^3-1)(y+9D^3-3Dx-36)}{(x+9D^2)^3+(3D^3-Dx-12)^3}$ *and* $\xi = \frac{12(D^3-1)}{Du+v+1}$.

Proof. Sending the point $\boldsymbol{P_0} = (0, -1)$ to the origin via the map $v \mapsto v - 1$, Eq. (3) becomes

$$\sum_{i=1}^{3} c_i(u, v) = 0 \tag{*}$$

where $c_3(u, v) = u^3 + v^3$, $c_2(u, v) = -3v(Du + v)$ and $c_1(u, v) = 3(Du + v)$. The slope λ of the tangent at $\boldsymbol{P_0}$ is equal to $-D$. Letting $d(u, v) = c_2(u, v)^2 - 4c_1(u, v)c_3(u, v)$, we have $d(u, \lambda u+1) = 12(D^3-1)u^3-27D^2u^2+18Du-3$. Hence, by Nagell reduction (see Theorem 7.4.9 in [5, p. 393]) and letting $B = 12(D^3-1)$,

Eq. (*) is birationnally equivalent to $y^2 = x^3 - 27D^2x^2 + 18DBx - 3B^2$ under the transformations

$$\begin{aligned}(u,v) &= \left(\tfrac{x(By-c_2(x,\lambda x+B))}{2c_3(x,\lambda x+B)}, \tfrac{(\lambda x+B)(By-c_2(x,\lambda x+B))}{2c_3(x,\lambda x+B)}\right)\\ &= \left(\tfrac{Bx(y+3B-3Dx)}{2(x^3+(B-Dx)^3)}, \tfrac{B(B-Dx)(y+3B-3Dx)}{2(x^3+(B-Dx)^3)}\right)\end{aligned}$$

and, noting from Eq. (*) that $2c_3(u,v) + c_2(u,v) = -2c_1(u,v) - c_2(u,v)$,

$$(x,y) = \left(\tfrac{Bu}{v-\lambda u}, \tfrac{B(2c_3(u,v)+c_2(u,v))}{(v-\lambda u)^2}\right) = \left(\tfrac{12(D^3-1)u}{Du+v}, \tfrac{36(D^3-1)(v-2)}{Du+v}\right) .$$

Replacing now x by $x+9D^2$, we finally obtain the required equation and the corresponding transformations. □

A 'straight-forward' application of the chord-and-tangent rule yields rather cumbersome formulæ for the doubling and the addition on an Hessian curve. The correct way is to use the Cauchy-Desboves' s formulæ (see Appendix A), which exploit the symmetry of Eq. (4). Plugging $W = 0$ into Eq. (4), we get the point at infinity $\boldsymbol{O} = (1 : -1 : 0)$. The inverse of $\boldsymbol{O}$ is $\boldsymbol{O}$. For $\boldsymbol{P} \neq \boldsymbol{O}$ we can work in affine coordinates. Let $\boldsymbol{P} = (u_1, v_1)$ be a point on the curve. The line $v = -u + (u_1 + v_1)$ contains the point $\boldsymbol{P}$ and, considering its projective version $V = -U + (u_1 + v_1)Z$, it also contains the point at infinity $\boldsymbol{O}$. Therefore, $-\boldsymbol{P}$ is the third point of intersection of this line connecting $\boldsymbol{P}$ and $\boldsymbol{O}$ with the curve. Substituting $v = -u + (u_1 + v_1)$ into Eq. (3), we obtain

$$\begin{aligned}& u^3 + (-u + (u_1 + v_1))^3 + 1 = 3Du(-u + (u_1 + v_1))\\ \iff\ & 3(u_1 + v_1 + D)u^2 - 3(u_1 + v_1)(u_1 + v_1 + D)u + (u_1 + v_1)^3 + 1 = 0\\ \iff\ & u^2 - (u_1 + v_1)u + u_1v_1 = 0 .\end{aligned}$$

(Note that $u_1 + v_1 + D \neq 0$ because $D^3 \neq 1$.) So, the u-coordinate of $-\boldsymbol{P}$ is v_1, and hence its v-coordinate is u_1, i.e., $-\boldsymbol{P} = (v_1, u_1)$ or, in projective coordinates,

$$-\boldsymbol{P} = (V_1 : U_1 : W_1) . \tag{8}$$

We use the same notations as in Appendix A. The tangent at $\boldsymbol{P} = (U_1 : V_1 : W_1)$ intersects the curve at the third point $-2\boldsymbol{P}$ whose coordinates are given by Eq. (14), with $F(U,V,W) = U^3 + V^3 + W^3 - 3DUVW$. We have $\varphi = 3({U_1}^2 - DV_1W_1)$, $\chi = 3({V_1}^2 - DU_1W_1)$ and $\psi = 3({W_1}^2 - DU_1V_1)$ and so, $-2\boldsymbol{P} = ((\psi^3 - \chi^3)/{U_1}^2 : (-\psi^3 + \varphi^3)/{V_1}^2 : (\chi^3 - \varphi^3)/{W_1}^2)$. A short calculation gives

$$\begin{aligned}\psi^3 - \chi^3 &= 27({W_1}^2 - DU_1V_1)^3 - 27({V_1}^2 - DU_1W_1)^3\\ &= 27[({W_1}^6 - {V_1}^6) + D^3{U_1}^3({W_1}^3 - {V_1}^3) - 3DU_1V_1W_1({W_1}^3 - {V_1}^3)]\\ &= 27({W_1}^3 - {V_1}^3)(D^3 - 1){U_1}^3 ,\end{aligned}$$

and, by symmetry, $-\psi^3 + \varphi^3 = 27({U_1}^3 - {W_1}^3)(D^3 - 1){V_1}^3$ and $\chi^3 - \varphi^3 = 27({V_1}^3 - {U_1}^3)(D^3 - 1){W_1}^3$. Hence, with Eq. (8), we finally obtain

$$2\boldsymbol{P} = \big(V_1({U_1}^3 - {W_1}^3) : U_1({W_1}^3 - {V_1}^3) : W_1({V_1}^3 - {U_1}^3)\big) . \tag{9}$$

From Eq. (15) (in Appendix A), the line connecting the points $\boldsymbol{P} = (U_1 : V_1 : W_1)$ and $\boldsymbol{Q} = (U_2 : V_2 : W_2)$ intersects the curve at the third point $-(\boldsymbol{P}+\boldsymbol{Q}) = (U_1\Theta - U_2\Upsilon : V_1\Theta - V_2\Upsilon : W_1\Theta - W_2\Upsilon)$, where $\Theta = 3U_1({U_2}^2 - DV_2W_2) + 3V_1({V_2}^2 - DU_2W_2) + 3W_1({W_2}^2 - DU_2V_2)$ and $\Upsilon = 3U_2({U_1}^2 - DV_1W_1) + 3V_2({V_1}^2 - DU_1W_1) + 3W_2({W_1}^2 - DU_1V_1)$. We have

$$\begin{aligned}
U_1\Theta - U_2\Upsilon &= 3V_1V_2(U_1V_2 - U_2V_1) \\
&\quad + 3W_1W_2(U_1W_2 - U_2W_1) - 3D({U_1}^2V_2W_2 - {U_2}^2V_1W_1)\,, \\
V_1\Theta - V_2\Upsilon &= 3U_1U_2(U_2V_1 - U_1V_2) \\
&\quad + 3W_1W_2(V_1W_2 - V_2W_1) - 3D({V_1}^2U_2W_2 - {V_2}^2U_1W_1)\,, \\
W_1\Theta - W_2\Upsilon &= 3U_1U_2(U_2W_1 - U_1W_2) \\
&\quad + 3V_1V_2(V_2W_1 - V_1W_2) - 3D({W_1}^2U_2V_2 - {W_2}^2U_1V_1)\,,
\end{aligned}$$

and thus, exploiting the fact that $\boldsymbol{P}$ and $\boldsymbol{Q}$ belong to the curve [9, no. 12], we obtain

$$\begin{aligned}
\frac{U_1\Theta - U_2\Upsilon}{W_1\Theta - W_2\Upsilon} &= \frac{{U_1}^2V_2W_2 - {U_2}^2V_1W_1}{{W_1}^2U_2V_2 - {W_2}^2U_1V_1}\,, \\
\frac{V_1\Theta - V_2\Upsilon}{W_1\Theta - W_2\Upsilon} &= \frac{{V_1}^2U_2W_2 - {V_2}^2U_1W_1}{{W_1}^2U_2V_2 - {W_2}^2U_1V_1}\,.
\end{aligned}$$

Therefore, with Eq. (8), the sum $\boldsymbol{R} = \boldsymbol{P} + \boldsymbol{Q}$ is given by

$$\boldsymbol{R} = \left({V_1}^2U_2W_2 - {V_2}^2U_1W_1 : {U_1}^2V_2W_2 - {U_2}^2V_1W_1 : {W_1}^2U_2V_2 - {W_2}^2U_1V_1\right)\,. \quad (10)$$

We now study the points of order 2 and 3. We work in affine coordinates since we are looking at points $\boldsymbol{P} \neq \boldsymbol{O}$ such that $2\boldsymbol{P} = \boldsymbol{O}$ or $3\boldsymbol{P} = \boldsymbol{O}$. Let $\boldsymbol{P} = (u_1, v_1)$. The condition $2\boldsymbol{P} = \boldsymbol{O}$ is equivalent to $\boldsymbol{P} = -\boldsymbol{P}$. Therefore, since $-\boldsymbol{P} = (v_1, u_1)$, the points $\boldsymbol{P} = (u_1, v_1)$ of order 2 are those for which $u_1 = v_1$.

Suppose $\boldsymbol{P} = (u_1, v_1)$ with $u_1 \neq v_1$, that is, $\boldsymbol{P}, 2\boldsymbol{P} \neq \boldsymbol{O}$. To find the points $\boldsymbol{P}$ of order 3, we use the doubling formula: $3\boldsymbol{P} = \boldsymbol{O} \iff 2\boldsymbol{P} = -\boldsymbol{P}$. So, a few algebra shows that the points of order 3 are exactly those with $u_1 = 0$ or $v_1 = 0$. In particular, the points $(0, -1)$ and $(-1, 0)$ have order 3.

Finally, it is interesting to note that a generic point $\boldsymbol{P} = (U : V : W)$ on the Hessian curve (4) satisfies

$$(D^2 + D + 1)(U + V + W)^3 = 3(DU + V + W)(U + DV + W)(U + V + DW)$$

since $(D^2+D+1)(U+V+W)^3 - 3(DU+V+W)(U+DV+W)(U+V+DW) = (D-1)^2(U^3+V^3+W^3-3DUVW) = 0$. Moreover, since $(DU+V+W)+(U+DV+W)+(U+V+DW) = (D+2)(U+V+W)$, it follows that

$$(\tilde{U} + \tilde{V} + \tilde{W})^3 = 3\tilde{D}\tilde{U}\tilde{V}\tilde{W}\,, \quad (11)$$

where $\begin{cases} \tilde{U} = DU + V + W \\ \tilde{V} = U + DV + W \\ \tilde{W} = U + V + DW \end{cases}$ and $\tilde{D} = \frac{(D+2)^3}{D^2+D+1}$.

4 Side-Channel Attacks

At CRYPTO '96 and subsequently at CRYPTO '99, Kocher *et al.* introduced a new class of attacks, the so-called *side-channel attacks*. By measuring some side-channel information (e.g., timing [11], power consumption [12]), they were able to find the secret keys from tamper-resistant devices.

When only a single measurement is performed the attack is referred to as *simple side-channel attack*, and when there are several correlated measurements sometimes it is referred to as a *differential side-channel attack*. The main concern at the moment for public-key cryptography are the simple side-channel attacks [12]. Efficient countermeasures are known for exponentiation-based cryptosystems (e.g., [4]), but they require the atomic operations to be indistinguishable. For elliptic curve cryptography, the atomic operations are addition, subtraction and doubling of points. Within the Weierstraß model, as suggested in [1], these operations appear to be different and some secret information may therefore leak through side-channel analysis.

The next section shows that the Hessian parameterization allows one to implement the same algorithm for the addition (or subtraction) of two points or for the doubling of a point.

5 Implementing the Hessian Curves

Figure 1 gives a detailed implementation to add two (different) points on an Hessian curve. The algorithm requires 12 multiplications (or 10 multiplications if one point has its last coordinate equal to 1) and 7 temporary variables.

We note that there are variants for this implementation. For instance, we are able to describe similar implementations with only 4 auxiliary variables and 18 multiplications, 5 auxiliary variables and 16 multiplications, and 6 auxiliary variables and 14 multiplications.

More remarkably, owing to the high symmetry of the Hessian parameterization, the *same* algorithm can be used for doubling a point. We have:

Proposition 2. *Let* $\boldsymbol{P} = (U_1 : V_1 : W_1)$ *be a point on an Hessian elliptic curve* $E_D(\mathbb{K})$*. Then*

$$2(U_1 : V_1 : W_1) = (W_1 : U_1 : V_1) + (V_1 : W_1 : U_1) \ . \tag{12}$$

Furthermore, we have $(W_1 : U_1 : V_1) \neq (V_1 : W_1 : U_1)$.

Proof. Addition formula (10) yields $(W_1 : U_1 : V_1)+(V_1 : W_1 : U_1) = ({U_1}^2V_1U_1 - {W_1}^2W_1V_1 : {W_1}^2W_1U_1 - {V_1}^2U_1V_1 : {V_1}^2V_1W_1 - {U_1}^2W_1U_1) = (V_1({U_1}^3 - {W_1}^3) : U_1({W_1}^3 - {V_1}^3) : W_1({V_1}^3 - {U_1}^3)) = 2(U_1 : V_1 : W_1)$ by Eq. (9).

```
Input:  P = (U1 : V1 : W1) and Q = (U2 : V2 : W2) with P ≠ Q
Output: P + Q = (U3 : V3 : W3)
  T1 ← U1; T2 ← V1; T3 ← W1 T4 ← U2; T5 ← V2; T6 ← W2
  T7 ← T1 · T6  (= U1W2)
  T1 ← T1 · T5  (= U1V2)
  T5 ← T3 · T5  (= W1V2)
  T3 ← T3 · T4  (= W1U2)
  T4 ← T2 · T4  (= V1U2)
  T2 ← T2 · T6  (= V1W2)
  T6 ← T2 · T7  (= U1V1W2^2)
  T2 ← T2 · T4  (= V1^2U2W2)
  T4 ← T3 · T4  (= V1W1U2^2)
  T3 ← T3 · T5  (= W1^2U2V2)
  T5 ← T1 · T5  (= U1W1V2^2)
  T1 ← T1 · T7  (= U1^2V2W2)
  T1 ← T1 − T4; T2 ← T2 − T5; T3 ← T3 − T6
  U3 ← T2; V3 ← T1; W3 ← T3
```

Fig. 1. AddHesse($\boldsymbol{P}$, $\boldsymbol{Q}$): Addition algorithm on an Hessian curve.

The second part of the proposition follows by contradiction. Suppose that $(W_1 : U_1 : V_1) = (V_1 : W_1 : U_1)$, i.e., that there exists some $t \in \mathbb{K}^*$ s.t. $W_1 = tV_1$, $U_1 = tW_1$ and $V_1 = tU_1$. This implies $W_1 \neq 0$ and $t^3 = 1$. Moreover, since $(U_1 : V_1 : W_1) \in E_D(\mathbb{K})$, ${U_1}^3 + {V_1}^3 + {W_1}^3 = 3DU_1V_1W_1$, which in turn implies $(t^3 + t^6 + 1){W_1}^3 = 3Dt^3{W_1}^3$ and thus $D = 1$, a contradiction by Lemma 1. □

In [13], Liardet and Smart suggest to represent elliptic curves as the intersection of two quadrics in $\mathbb{P}^3$ as a means to protect against side-channel attacks. Considering the special case of an elliptic curve whose order is divisible by 4 (i.e., the Jacobi form), they observe that the same algorithm can be used for adding and doubling points with **16 multiplications** (see also [3] for the formulæ). Using the proposed Hessian parameterization, only **12 multiplications** are necessary for adding or doubling points. The Hessian parameterization gives thus a **33% improvement** over the Jacobi parameterization. Another advantage of the Hessian parameterization is that points are represented with fewer coordinates, which results in substantial memory savings.

Finally, contrary to other parameterizations, there is no (field) subtraction to compute the inverse of a point (see Eq. (8)). Hence, our addition algorithm can be used *as is* for subtracting two points $\boldsymbol{P} = (U_1 : V_1 : W_1)$ and $\boldsymbol{Q} = (U_2 : V_2 : W_2)$ on an Hessian elliptic curve:

$$(U_1 : V_1 : W_1) - (U_2 : V_2 : W_2) = (U_1 : V_1 : W_1) + (V_2 : U_2 : W_2) \ . \qquad (13)$$

To sum up, by adapting the order of the inputs accordingly to Eq. (12) or (13), the addition algorithm presented in Fig. 1 can be used *indifferently* for

– adding two (different) points;

– doubling a point;
– subtracting two points;

with only 12 multiplications and 7 auxiliary variables including the 3 result variables. This results in the *fastest* known method for implementing the elliptic curve scalar multiplication towards resistance against side-channel attacks.

Acknowledgements. The first author would like to acknowledge Prof. Laih and the members of his lab for their generous hospitality while part of this work was performed. Thanks also to the anonymous referees.

References

1. IEEE Std 1363-2000, *IEEE standard specifications for public-key cryptography*, IEEE Computer Society, August 29, 2000.
2. J. W. S. Cassels, *Lectures on elliptic curves*, London Mathematical Society Student Texts, vol. 24, Cambridge University Press, 1991.
3. D. V. Chudnovsky and G. V. Chudnovsky, *Sequences of numbers generated by addition in formal groups and new primality and factorization tests*, Advances in Applied Math. **7** (1986/7), 385–434.
4. Christophe Clavier and Marc Joye, *Universal exponentiation algorithm: A first step towards provable SPA-resistance*, these proceedings.
5. Henri Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, vol. 138, Springer-Verlag, 1993.
6. Henri Cohen, Atsuko Miyaji, and Takatoshi Ono, *Efficient elliptic curve exponentiation using mixed coordinates*, Advances in Cryptology – ASIACRYPT '98 (K. Ohta and D. Pei, eds.), Lecture Notes in Computer Science, vol. 1514, Springer-Verlag, 1998, pp. 51–65.
7. Jean-Sébastien Coron, *Resistance against differential power analysis for elliptic curve cryptosystems*, Cryptographic Hardware and Embedded Systems (CHES '99) (Ç.K. Koç and C. Paar, eds.), Lecture Notes in Computer Science, vol. 1717, Springer-Verlag, 1999, pp. 292–302.
8. Erik De Win, Serge Mister, Bart Preneel, and Michael Wiener, *On the performance of signature schemes based on elliptic curves*, Algorithmic Number Theory Symposium (J.-P. Buhler, ed.), Lecture Notes in Computer Science, vol. 1423, Springer-Verlag, 1998, pp. 252–266.
9. M. Desboves, *Résolution, en nombres entiers et sous sa forme la plus générale, de l'équation cubique, homogène, à trois inconnues*, Ann. de Mathémat. **45** (1886), 545–579.
10. Otto Hesse, *Über die Elimination der Variabeln aus drei algebraischen Gleichungen vom zweiten Grade mit zwei Variabeln*, Journal für die reine und angewandte Mathematik **10** (1844), 68–96.
11. Paul C. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, Advances in Cryptology – CRYPTO '96 (N. Koblitz, ed.), Lecture Notes in Computer Science, vol. 1109, Springer-Verlag, 1996, pp. 104–113.
12. Paul Kocher, Joshua Jaffe, and Benjamin Jun, *Differential power analysis*, Advances in Cryptology – CRYPTO '99 (M. Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1999, pp. 388–397.

13. Pierre-Yvan Liardet and Nigel P. Smart, *Preventing SPA/DPA in ECC systems using the Jacobi form*, these proceedings.
14. Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan, *Power analysis attacks of modular exponentiation in smartcards*, Cryptographic Hardware and Embedded Systems (CHES '99) (Ç.K. Koç and C. Paar, eds.), Lecture Notes in Computer Science, vol. 1717, Springer-Verlag, 1999, pp. 144–157.
15. Nigel P. Smart, *The Hessian form of an elliptic curve*, these proceedings.

A Cauchy-Desboves' s Formulæ

Let $F(U,V,W)=0$ be the (homogeneous) equation of a general cubic curve and let $\boldsymbol{P_1}=(U_1:V_1:W_1)$ and $\boldsymbol{P_2}=(U_2:V_2:W_2)$ be two points on the curve.

We let denote $\varphi=\frac{\partial F(\boldsymbol{P_1})}{\partial U}$, $\chi=\frac{\partial F(\boldsymbol{P_1})}{\partial V}$ and $\psi=\frac{\partial F(\boldsymbol{P_1})}{\partial W}$. Then the tangent at $\boldsymbol{P_1}$ intersects the curve at the third point [9, Eq. (16)] given by

$$\left(\frac{F(0,\psi,-\chi)}{{U_1}^2}:\frac{F(-\psi,0,\varphi)}{{V_1}^2}:\frac{F(\chi,-\varphi,0)}{{W_1}^2}\right)\ . \tag{14}$$

Moreover, the secant joining $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ intersects the curve at the third point [9, Eq. (16)] given by

$$(U_1\Theta-U_2\Upsilon:V_1\Theta-V_2\Upsilon:W_1\Theta-W_2\Upsilon)\ , \tag{15}$$

where $\Theta=U_1\frac{\partial F(\boldsymbol{P_2})}{\partial U}+V_1\frac{\partial F(\boldsymbol{P_2})}{\partial V}+W_1\frac{\partial F(\boldsymbol{P_2})}{\partial W}$ and $\Upsilon=U_2\frac{\partial F(\boldsymbol{P_1})}{\partial U}+V_2\frac{\partial F(\boldsymbol{P_1})}{\partial V}+W_2\frac{\partial F(\boldsymbol{P_1})}{\partial W}$.

B Samples

Here are two examples of cryptographic Hessian elliptic curves $E_D(\mathbb{F}_p)$ defined over the prime field $\mathbb{F}_p$ with $p=2^{160}-2933$ and $p=2^{224}-2^{10}-1$, respectively. Both curves are adapted from [6] using Proposition 1. Note that since $(0,-1)$ is on the curve whatever the values of D and p and that this point has order 3, the order of an Hessian curve, $\#E_D(\mathbb{F}_p)$, is always a multiple of 3. Note also that this specialized representation does not impact the security of the resulting cryptographic applications.

B.1 160-bit prime

$p=2^{160}-2933$
$D=945639186043697550302587435415597619883075636292$
$\#E_D(\mathbb{F}_p)=3\cdot5\cdot157\cdot6205951750874322370291655293816111692249133 37$

B.2 224-bit prime

$p=2^{224}-2^{10}-1$
$D=25840187014857916932759133078916563544400020237401312879815735566345$
$\#E_D(\mathbb{F}_p)=3\cdot23\cdot3907238647413136202125654360437627777128235166 7\backslash$
3432244734573782061

Protections against Differential Analysis for Elliptic Curve Cryptography
– An Algebraic Approach –

[Published in Ç.K. Koç, D. Naccache, and C. Paar, Eds., *Cryptographic Hardware and Embedded Systems – CHES 2001*, vol. 2162 of *Lecture Notes in Computer Science*, pp. 377–390, Springer-Verlag, 2001.]

Marc Joye[1] and Christophe Tymen[2]

[1] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
marc.joye@gemplus.com – http://www.geocities.com/MarcJoye/
[2] École Normale Supérieure
45 rue d'Ulm, 75230 Paris, France
christophe.tymen@gemplus.com

Abstract. We propose several new methods to protect the scalar multiplication on an elliptic curve against Differential Analysis. The basic idea consists in transforming the curve through various random morphisms to provide a non-deterministic execution of the algorithm.
The solutions we suggest complement and improve the state-of-the-art, but also provide a practical toolbox of efficient countermeasures. These should suit most of the needs for protecting implementations of crypto-algorithm based on elliptic curves.

Keywords. Public-key cryptography, side-channel attacks, timing attacks, differential power analysis (DPA), elliptic curves, smart cards.

1 Introduction

Since the introduction of the timing attacks [10] by Paul Kocher in 1996 and subsequently of the Differential Power Analysis (DPA) [9], the so-called side-channel attacks have become a major threat against tamper-resistant devices like smart-cards, to the point where the immediate relevance of classical security notions is somewhat questionable. Furthermore, numerous experiments show that most of the time, perfunctory countermeasures do not suffice to thwart those attacks.

In the case of public-key cryptosystems based on the discrete logarithm on elliptic curves, the running time does not really represent a bottleneck for smart-card applications, which are equipped with additional devices for fast computation in finite fields. Therefore, investigating the security of these applications,

less constrained by performance criteria, against side-channel attacks is very relevant.

Compared to the previous works of [5] and [7], this paper systematically develops the same idea: assuming that an elliptic curve cryptosystem executes some operations in the group of a curve E, the whole algorithm is transposed to a curve $\phi(E)$, where ϕ is a random morphism. The rich algebraic structure of elliptic curves enables numerous possible choices for such morphisms.

The rest of this paper is organized as follows. In the next section, we provide a brief description of elliptic curves. We refer the reader to Appendix A for further mathematical details. The general principles of differential analysis and how this can reveal the secret keys of an elliptic curve cryptosystem are explained in Section 3. Next, we provide two main classes of possible morphisms to randomize the basepoint. Finally, we present in Section 5 a new randomization of the encoding of the multiplier in the case of Anomalous Binary Curves (ABC).

2 Elliptic Curves

Let $\mathbb{K}$ be a field. An *elliptic curve over* $\mathbb{K}$ is a pair $(E, \boldsymbol{O})$ where E is a non-singular curve of genus one over $\mathbb{K}$ with a point $\boldsymbol{O} \in E$. It is well known that the set of points $(x, y) \in \mathbb{K} \times \mathbb{K}$ verifying the (non-singular) Weierstraß equation

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (a_i \in \mathbb{K}) \tag{1}$$

together with $\boldsymbol{O}$ form an elliptic curve and that an elliptic curve can always be expressed in such a form. The point $\boldsymbol{O}$ is called the *point at infinity*.

The set of points (x, y) satisfying Eq. (1) and $\boldsymbol{O}$ form an Abelian group where $\boldsymbol{O}$ is the neutral element. This group is denoted by $E(\mathbb{K})$ and the group operation is denoted by $+$. The operation consisting in computing the multiple of a point, $\boldsymbol{Q} = k\boldsymbol{P} := \boldsymbol{P} + \cdots + \boldsymbol{P}$ (k times), is called *(elliptic curve) scalar multiplication*. We refer the unfamiliar reader to Appendix A for the required background on this particular topic.

3 Differential Analysis

In his CRYPTO '96 paper [10] and thereafter in [9] with Jaffe and Jun, Kocher launched a new class of attacks, the so-called *side-channel attacks*.

The basic idea of the side-channel attacks is that some side-channel information (e.g., timing, power consumption, electromagnetic radiation) of a device depends on the operations it performs. For instance, it is well known that the modification of a memory state yields a different power consumption according to the memory goes from one to zero, or the opposite.

By capturing this information, it may be possible to recover some secret keys involved during the execution of a crypto-algorithm, at least in a careless implementation. When a single input is used in eliciting information, the process is

referred to as a *Simple Analysis* and when there are several inputs used together with statistical tools, it is referred to as a *Differential Analysis*. In this paper, we are concerned with the second type of attack, and in particular in the context of elliptic curve cryptography.

For elliptic curve cryptosystems, this type of attack applies to the scalar multiplication. Following [5], a simple countermeasure to defeat simple analysis attacks resides in replacing the standard double-and-add algorithm by a double-and-add-*always* algorithm for computing $\boldsymbol{Q} = k\boldsymbol{P}$ on an elliptic curve (see also [7] for further countermeasures dedicated to ABC curves). However, such an algorithm is still susceptible to a differential analysis attack. Let $k = (k_{m-1}, \ldots, k_0)_2$ be the binary expansion of multiplier k. Suppose that an attacker already knows the highest bits, $k_{m-1}, \ldots, k_{j+1}$, of k. Then, he *guesses* that the next bit k_j is equal to one. He randomly chooses several points $\boldsymbol{P_1}, \ldots, \boldsymbol{P_t}$ and computes $\boldsymbol{Q_r} = (\sum_{i=j}^{m-1} k_i)\boldsymbol{P_r}$ for $1 \leq r \leq t$. Using a boolean selection function g, he prepares two sets: the first set, $\mathcal{S}_{\texttt{true}}$, contains the points $\boldsymbol{P_r}$ such that $g(\boldsymbol{Q_r}) = \texttt{true}$ and the second set, $\mathcal{S}_{\texttt{false}}$, contains those such that $g(\boldsymbol{Q_r}) = \texttt{false}$. Depending on the side-channel information monitored by the attacker and the actual implementation, a selection function may, for example, be the value of a given bit in the representation of $\boldsymbol{Q_r}$.

Let $C(r)$ denote the side-channel information associated to the computation of $k\boldsymbol{P_r}$ by the cryptographic device (e.g., the power consumption). If the guess $k_j = 1$ is incorrect then the difference

$$\langle C(r) \rangle_{\substack{1 \leq r \leq t \\ \boldsymbol{P_r} \in \mathcal{S}_{\texttt{true}}}} - \langle C(r) \rangle_{\substack{1 \leq r \leq t \\ \boldsymbol{P_r} \in \mathcal{S}_{\texttt{false}}}}$$

will be ≈ 0 as the two sets appear as two random (i.e. uncorrelated) sets; otherwise the guess is correct. Once k_j is known, the remaining bits, $k_{j-1}, \ldots, k_0$, are recovered recursively, in the same way. We note that such attacks are not restricted to binary methods and can be adapted to work with other scalar multiplication methods, as well.

To thwart differential attacks, it is recommended to randomize the basepoint $\boldsymbol{P}$ and the multiplier k in the computation of $\boldsymbol{Q} = k\boldsymbol{P}$. Several countermeasures are already known. See [5] for general curves and [7] for ABC curves. The next section proposes two techniques for randomizing the basepoint and Section 5 shows how to randomize the multiplier for an ABC curve.

4 Randomizing the Basepoint

4.1 Elliptic curve isomorphisms

We first recall some results on isomorphisms between elliptic curves. We say that two elliptic curves over a field $\mathbb{K}$ defined by their Weierstraß equations E and E' are *isomorphic over* $\mathbb{K}$ (or $\mathbb{K}$-*isomorphic*) if they are isomorphic as projective varieties. It turns out that curve isomorphisms induce group morphisms. The determination of isomorphisms between two given elliptic curves is solved in the next two corollaries.

Corollary 1. *Let* $\mathbb{K}$ *be a field with* $\operatorname{Char}\mathbb{K} \neq 2, 3$. *The elliptic curves given by* $E_{/\mathbb{K}} : y^2 = x^3 + ax + b$ *and* $E'_{/\mathbb{K}} : y^2 = x^3 + a'x + b'$ *are* $\mathbb{K}$*-isomorphic if and only if there exists* $u \in \mathbb{K}^*$ *such that* $u^4a' = a$ *and* $u^6b' = b$. *Furthermore, we have*

$$\varphi : E(\mathbb{K}) \xrightarrow{\sim} E'(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (u^{-2}x, u^{-3}y) \end{cases} \tag{2}$$

and

$$\varphi^{-1} : E'(\mathbb{K}) \xrightarrow{\sim} E(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (u^2x, u^3y) \end{cases} . \tag{3}$$

Proof. With the notations of Proposition 2 (in appendix), we obtain $r = s = t = 0$ and so $u^4a'_4 = a_4$ and $u^6a'_6 = a_6 \iff u^4a' = a$ and $u^6b' = b$ for some $u \in \mathbb{K}^*$. □

Corollary 2. *Let* $\mathbb{K}$ *be a field with* $\operatorname{Char}\mathbb{K} = 2$. *The (non-supersingular) elliptic curves given by* $E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b$ *and* $E'_{/\mathbb{K}} : y^2 + xy = x^3 + a'x^2 + b'$ *are* $\mathbb{K}$*-isomorphic if and only if there exists* $s \in \mathbb{K}$ *such that* $a' = a + s + s^2$ *and* $b' = b$. *Furthermore, we have*

$$\varphi : E(\mathbb{K}) \xrightarrow{\sim} E'(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (x, y + sx) \end{cases} \tag{4}$$

and

$$\varphi^{-1} : E'(\mathbb{K}) \xrightarrow{\sim} E(\mathbb{K}), \begin{cases} \boldsymbol{O} \mapsto \boldsymbol{O} \\ (x, y) \mapsto (x, y + sx) \end{cases} . \tag{5}$$

Proof. From Proposition 2, the relation $ua'_1 = a_1 + 2s$ gives $u = 1$. The third and fourth relations give $r = 0$ and $t = 0$, respectively. Hence, from the second relation we have $a'_2 = a_2 - s - s^2$ whereas the last one yields $a'_6 = a_6 \iff a' = a + s + s^2$ and $b' = b$. □

We can thus randomize the scalar multiplication algorithm as follows. We perform the scalar multiplication on a random isomorphic elliptic curve and then we come back to the original elliptic curve. More formally, if φ is a random isomorphism from $E_{/\mathbb{K}}$ to $E'_{/\mathbb{K}}$, we propose to compute $\boldsymbol{Q} = k\boldsymbol{P}$ in $E(\mathbb{K})$ according to

$$\boldsymbol{Q} = \varphi^{-1}\big(k\big(\varphi(\boldsymbol{P})\big)\big)\,, \tag{6}$$

or schematically,

$$\begin{array}{ccc} \boldsymbol{P} \in E(\mathbb{K}) & \xrightarrow{\text{mult. by } k \text{ map}} & \boldsymbol{Q} = k\boldsymbol{P} \in E(\mathbb{K}) \\ \varphi \big\downarrow & & \big\uparrow \varphi^{-1} \\ \boldsymbol{P}' \in E'(\mathbb{K}) & \xrightarrow{\text{mult. by } k \text{ map}} & \boldsymbol{Q}' = k\boldsymbol{P}' \in E'(\mathbb{K}) \end{array}$$

Corollaries 1 and 2 indicate that computing the image of a point through an elliptic curve isomorphism can be done using only a few elementary field

operations. This yields a very efficient means to randomize the computation of $\boldsymbol{Q} = k\boldsymbol{P}$.

Algorithm 1 (Scalar Multiplication via Random Isomorphic Elliptic Curves for Char $\mathbb{K} \neq 2, 3$).

Input: A point $\boldsymbol{P} = (x_1, y_1) \in E(\mathbb{K})$ with $E_{/\mathbb{K}} : y^2 = x^3 + ax + b$.
An integer k.
Output: The point $\boldsymbol{Q} = k\boldsymbol{P}$.

1. Randomly choose an element $u \in \mathbb{K}^*$;
2. Form the point $\boldsymbol{P'} \leftarrow (u^{-2}\, x_1, u^{-3}\, y_1)$;
3. Evaluate $a' \leftarrow u^{-4}\, a$;
4. Compute $\boldsymbol{Q'} \leftarrow k\boldsymbol{P'}$ in $E'(\mathbb{K})$ with $E'_{/\mathbb{K}} : y^2 = x^3 + a'x + b'$;[1]
5. If $(\boldsymbol{Q'} = \boldsymbol{O})$ then return $\boldsymbol{Q} = \boldsymbol{O}$ and stop. Otherwise set $\boldsymbol{Q'} \leftarrow (x'_3, y'_3)$;
6. Return $\boldsymbol{Q} = (u^2\, x'_3, u^3\, y'_3)$.

In [5, § 5.3], Coron suggests the randomization of projective coordinates in order to blind the basepoint $\boldsymbol{P}$: $\boldsymbol{P} = (x_1, y_1)$ is randomized into $(t^2x_1 : t^3y_1 : t)$ in Jacobian coordinates (or into $(tx_1 : ty_1 : t)$ in homogeneous coordinates) for some $t \in \mathbb{K}^*$. The advantage of the proposed countermeasure is that, in Step 2 of Algorithm 1, we can represent $\boldsymbol{P'}$ as the projective point $\boldsymbol{P'} = (u^{-2}x_1 : u^{-3}y_1 : 1)$, that is, a point with its Z-coordinate equal to 1. This results in a faster scalar multiplication algorithm. Using the values of Table 2, we precisely quantify the number of (field) multiplications needed to compute $\boldsymbol{Q} = k\boldsymbol{P}$, considering in each case the faster coordinate system. This is summarized in the next table.[2]

Table 1. Average number of (field) multiplications to compute $\boldsymbol{Q} = k\boldsymbol{P}$.

	Random. proj. coord. ([5])				Algorithm 1	
	$a \neq -3$		$a = -3$			
Double-and-add	$17\frac{1}{2} \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$	$16 \cdot \lvert k\rvert_2$	$(\mathcal{J})$	$15 \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$
Double-and-add-or-sub.	$15\frac{1}{3} \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$	$13\frac{1}{3} \cdot \lvert k\rvert_2$	$(\mathcal{J})$	$12\frac{2}{3} \cdot \lvert k\rvert_2$	$(\mathcal{J}^m)$
Double-and-add-always	$25 \cdot \lvert k\rvert_2$	$(\mathcal{J}^c)$	$23 \cdot \lvert k\rvert_2$	$(\mathcal{J}^c)$	$21 \cdot \lvert k\rvert_2$	$(\mathcal{J})$

For fields of characteristic 2, random isomorphisms of elliptic curves cannot be considered *alone* as a means to protect against differential analysis. The x-coordinate of basepoint $\boldsymbol{P}$ remains invariant through isomorphism φ (cf. Eq. (4)) and so the resulting implementation may still be subject to a differential analysis

[1] Note that parameter b' is not required by the scalar multiplication algorithm.
[2] $\mathcal{J}$, $\mathcal{J}^c$ and $\mathcal{J}^m$ respectively refer to the Jacobian coordinates, Chudnovsky Jacobian coordinates and the modified Jacobian coordinates (see Appendix A.1).

attack. However, it can be combined with other countermeasures to offer an additional security level.

The next section presents a countermeasure that randomizes both the x- and the y-coordinates of point $\boldsymbol{P}$, whatever the characteristic of the field we are working with.

4.2 Field isomorphisms

Up to isomorphism, there is one and only one finite field L of characteristic p with p^n elements. Every such field may be viewed as the field generated (over $\mathbb{F}_p$) by a root of an irreducible monic polynomial Π of degree n. Given $\Pi(X)$, any element of L can be represented as a polynomial in $\mathbb{F}_p[X]/(\Pi)$. If $e \in L$, we note $\vartheta_\Pi(e)$ its corresponding representation in $\mathbb{F}_p[X]/(\Pi)$. From another irreducible monic polynomial $\Pi'(Y)$ of degree n, we obtain another representation for the elements $e \in L$, $\vartheta_{\Pi'}(e) \in \mathbb{F}_p[Y]/(\Pi')$. The fields $\mathbb{K} := \mathbb{F}_p[X]/(\Pi)$ and $\mathbb{K}' := \mathbb{F}_p[Y]/(\Pi')$ being isomorphic, we let ϕ denote such an isomorphism from $\mathbb{K}$ to $\mathbb{K}'$. The map ϕ extends to $\mathbb{K} \times \mathbb{K}$ with $\phi(x, y) = (\phi(x), \phi(y))$. In particular, ϕ transforms the equation of an elliptic curve over $\mathbb{K}$ into the equation of an elliptic curve over $\mathbb{K}'$, i.e.,

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

is transformed into

$$E'_{/\mathbb{K}'} : y^2 + \phi(a_1)xy + \phi(a_3)y = x^3 + \phi(a_2)x^2 + \phi(a_4)x + \phi(a_6) \ .$$

Consequently, isomorphisms between fields can be used to randomize the representation of the basepoint $\boldsymbol{P}$. To compute $\boldsymbol{Q} = k\boldsymbol{P}$, we first choose randomly a field $\mathbb{K}'$ isomorphic to $\mathbb{K}$ through isomorphism ϕ. Then, we compute $\boldsymbol{Q}$ as

$$\boldsymbol{Q} = \phi^{-1}\big(k\big(\phi(\boldsymbol{P})\big)\big) \ . \tag{7}$$

In other words, we represent $\boldsymbol{P} \in E(\mathbb{K})$ as a point $\boldsymbol{P'} \in E'(\mathbb{K}')$, next we compute $\boldsymbol{Q'} := k\boldsymbol{P'}$ in $E'(\mathbb{K}')$, and finally we go back to the original representation by representing $\boldsymbol{Q'}$ as a point $\boldsymbol{Q} \in E(\mathbb{K})$.

At first glance, it is unclear that this could lead to an countermeasure efficient in a constrained environment. Indeed, to build a field $\mathbb{K}'$ isomorphic to $\mathbb{K}$, a natural way consists in determining an irreducible monic polynomial of degree n, $\Pi' \in \mathbb{F}_p[Y]$. An isomorphism ϕ is then obtained by computing a root α of Π in $\mathbb{K}'$:

$$\phi : \mathbb{K} \xrightarrow{\sim} \mathbb{K}' : x \mapsto \sum_{i=0}^{n-1} x_i \, \alpha^i \, , \tag{8}$$

where $x = \sum_{i=0}^{n-1} x_i \, X^i \in \mathbb{K} = \mathbb{F}_p[X]/(\Pi)$. Likewise, the inverse map, from $\mathbb{K}'$ to $\mathbb{K}$, requires to find a root β of Π' in $\mathbb{K}$.

However, we can do much better when some permanent writable memory is at disposal (e.g., the EEPROM in a smart-card implementation). The general idea

is, given an isomorphism $\phi : \mathbb{K} \tilde{\to} \mathbb{K}'$ stored in EEPROM, to determine from ϕ and $\mathbb{K}'$ a new field $\mathbb{K}''$ and a new isomorphism $\phi' : \mathbb{K} \tilde{\to} \mathbb{K}''$, and so on. This can be done thanks to Proposition 1, which yields a recursive method for constructing irreducible polynomials of same degree.

Proposition 1. *Let T be a polynomial permutation[3] of $\mathbb{F}_{p^n}$ and let Π be an irreducible polynomial in $\mathbb{F}_p[X]$ of degree n. Then polynomial $\Pi \circ T$ has at least one irreducible factor of degree n, say Π', in $\mathbb{F}_p[X]$.*

Proof. Let α be a root of Π. As Π is irreducible, the orbit of α under the action of the Frobenius is of cardinality n. T being a permutation, the image of this orbit through T^{-1} is still of cardinality n. Since T is a polynomial with coefficients in $\mathbb{F}_p$, it commutes with the Frobenius, and thus the image of the orbit of α through T^{-1} appears as the orbit of $T^{-1}(\alpha)$. Consequently, the polynomial $\prod_i (X - (T^{-1}(\alpha))^{p^i})$ is irreducible of degree n, and divides $\Pi \circ T$. □

Hence, if we choose a polynomial permutation T of small degree (e.g., 2 or 3), we compute $\Pi \circ T$ and factor it with a *specific* algorithm to find Π'. A further generalization consists in storing a family of polynomial permutations $\mathfrak{S} = \{T_i\}$ in EEPROM and to randomly choose $T \in \mathfrak{S}$ when constructing Π'.

We note Π the publicly known polynomial which defines the field $\mathbb{K} = \mathbb{F}_p[X]/(\Pi)$ used as the reference field. We assume that another polynomial $\Pi^{(1)}$ defines the field $\mathbb{K}^{(1)}$ isomorphic to $\mathbb{K}$. We also assume that two polynomials $\alpha^{(1)}, \beta^{(1)} \in \mathbb{F}_p[X]$ of degree at most n verifying Eqs. (9) have initially been stored in EEPROM. These additional data must of course be kept secret. At the j^{th} execution of the scalar multiplication algorithm, the EEPROM contains an irreducible monic polynomial $\Pi^{(j)} \in \mathbb{F}_p[X]$ of degree n, and two polynomials $\alpha^{(j)}, \beta^{(j)} \in \mathbb{F}_p[X]$ such that

$$\begin{cases} \Pi(\beta^{(j)}) & \equiv 0 \pmod{\Pi^{(j)}} \\ \Pi^{(j)}(\alpha^{(j)}) & \equiv 0 \pmod{\Pi} \end{cases} . \tag{9}$$

These relations simply say that $\alpha^{(j)}$ and $\beta^{(j)}$ respectively define an isomorphism $\phi^{(j)}$ and its inverse from the field $\mathbb{K}$ to the field $\mathbb{F}_p[X]/(\Pi^{(j)})$ denoted by $\mathbb{K}^{(j)}$.

We are now ready to give the algorithm. We choose randomly $T \in \mathfrak{S}$ and determine an irreducible monic polynomial $\Pi^{(j+1)}$ of degree n in $\mathbb{F}_p[X]$ that divides $\Pi^{(j)} \circ T$ with a method that will be explained later. Then we set $\beta^{(j+1)} = \beta^{(j)} \circ T \bmod \Pi^{(j+1)}$, $\alpha^{(j+1)} = T^{-1}(\alpha^{(j)}) \bmod \Pi$, and we store $\alpha^{(j+1)}, \beta^{(j+1)}$ and $\Pi^{(j+1)}$ in EEPROM. Here, T^{-1} denotes the permutation inverse of T. It is easy to check that $\alpha^{(j+1)}, \beta^{(j+1)}$ and $\Pi^{(j+1)}$ still verify Eqs. (9), and thus define an isomorphism $\phi^{(j+1)}$ and its inverse from $\mathbb{K}$ to $\mathbb{F}_p[X]/(\Pi^{(j)})$. It remains to compute $\boldsymbol{P}' = \phi^{(j+1)}(\boldsymbol{P})$ and the coefficients of E'. Finally, we compute $k\boldsymbol{P}'$ in E' and convert the resulting point by the inverse isomorphism to obtain $\boldsymbol{Q} = k\boldsymbol{P}$. From the viewpoint of the running time, one of the advantages of this method is to skip the root finding step.

[3] A polynomial T with coefficients in $\mathbb{F}_p$ is a *polynomial permutation of* $\mathbb{F}_{p^n}$ if the map $x \mapsto T(x)$ permutes $\mathbb{F}_{p^n}$.

Algorithm 2 (Scalar Multiplication via Random Isomorphic Fields).

Input: A point $\boldsymbol{P} = (x_1, y_1) \in E(\mathbb{K})$
with $\begin{cases} E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b & \text{if } \mathrm{Char}\,\mathbb{K} = 2 \\ E_{/\mathbb{K}} : y^2 = x^3 + ax + b & \text{if } \mathrm{Char}\,\mathbb{K} > 3 \end{cases}$.
An integer k.
[EEPROM: Polynomials $\alpha^{(j)}$, $\beta^{(j)}$ and $\Pi^{(j)}$.]
Output: The point $\boldsymbol{Q} = k\boldsymbol{P}$.

1. Randomly choose $T \in \mathfrak{S}$;
2. Determine, in $\mathbb{F}_p[X]$, an irreducible monic polynomial $\Pi^{(j+1)}$ s.t.
$$\Pi^{(j+1)} \text{ divides } \Pi^{(j)} \circ T;$$
3. Set $\beta^{(j+1)} \leftarrow \beta^{(j)} \circ T \bmod \Pi^{(j+1)}$;
4. Set $\alpha^{(j+1)} \leftarrow T^{-1}(\alpha^{(j)}) \bmod \Pi$;
5. Update the EEPROM with $\alpha^{(j+1)}$, $\beta^{(j+1)}$ and $\Pi^{(j+1)}$;
6. Form $\boldsymbol{P'} \leftarrow \boldsymbol{P} \circ \beta^{(j+1)} \bmod \Pi^{(j+1)}$;
7. Evaluate $a' \leftarrow a \circ \beta^{(j+1)} \bmod \Pi^{(j+1)}$;
8. Compute $\boldsymbol{Q'} \leftarrow k\boldsymbol{P'}$ in $E'\big(\mathbb{F}_p[X]/(\Pi^{(j+1)})\big)$;
9. If $(\boldsymbol{Q'} = \boldsymbol{O})$ then return $\boldsymbol{Q} = \boldsymbol{O}$ and stop. Otherwise set $\boldsymbol{Q'} \leftarrow (x'_3, y'_3)$;
10. Return $\boldsymbol{Q} = (x'_3, y'_3) \circ \alpha^{(j+1)} \bmod \Pi$.

We still have to show how to solve Step 2 in the above algorithm. We illustrate the technique in the case $\mathbb{K} = \mathbb{F}_2[X]/(\Pi)$ with Π of degree n and $\gcd(2^n - 1, 3) = 1$ (this case includes the popular choice $n = 163$ for elliptic curve cryptosystems), but we stress that the proposed technique is fully general and can be adapted to the other cases, as well.

First, note that:

Lemma 1. *If* $\gcd(2^n - 1, 3) = 1$ *then the elements of*

$$\mathfrak{S} = \left\{X^3, 1 + X^3, X + X^2 + X^3, 1 + X + X^2 + X^3\right\} \subset \mathbb{F}_2[X]$$

permute $\mathbb{F}_{2^n}$.

Proof. Let α be a primitive element of $\mathbb{F}_{2^n}^*$ (remember that $\mathbb{F}_{2^n} = \mathbb{F}_{2^n}^* \cup \{0\}$). Then $\langle \alpha^3 \rangle$ generates a subgroup of order $(2^n - 1)/\gcd(2^n - 1, 3) = 2^n - 1$ and so α^3 is a primitive element or equivalently X^3 permutes $\mathbb{F}_{2^n}$. Suppose that there exist $\alpha, \beta \in \mathbb{F}_{2^n}$ s.t. $\alpha^3 + 1 = \beta^3 + 1 \iff \alpha^3 = \beta^3$. This implies $\alpha = \beta$ since X^3 is a permutation polynomial. The remaining cases are proved similarly by noting that $\alpha^2 - \beta^2 = (\alpha - \beta)^2$. □

Given a set $\mathfrak{S}$ of permutation polynomials, write $Q := \Pi^{(j)} \circ T$ for some $T \in \mathfrak{S}$ and Π irreducible of degree n. The fact that Q has degree $3n$ enables us to specialize the classical factorization algorithms (see, e.g., [3, p. 125]):

1. Compute $R = X^{2^n} - X \bmod Q$;
2. Then, using Proposition 1, $\Pi' = \gcd(Q, R)$ is irreducible of degree n in $\mathbb{F}_2[X]$.

5 Randomizing the Multiplier on ABC Curves

The other side of countermeasures for elliptic curve cryptography is the introduction of a random to blind the multiplier during the scalar multiplication. This technique is useful to prevent Differential Analysis, but may also contribute to an additional security level against Simple Analysis, as the multiplier is in general secret.

The proposed method is specific to ABC curves (see Appendix A.2 for the definitions) where the multiplier first goes through several encoding functions before being used in the scalar multiplication loop itself. We take advantage of the properties of this encoding to randomize the multiplier.

Building on previous works by Koblitz [8] and Meier-Staffelbach [11], Solinas presents in [14] a very efficient algorithm to compute $\boldsymbol{Q} = k\boldsymbol{P}$ on an ABC curve. Letting $\tau : (x, y) \mapsto (x^2, y^2)$ the Frobenius endomorphism, his algorithm proceeds as follows.

1. Compute, in $\mathbb{Z}[\tau]$, $\kappa \leftarrow k \bmod (\tau^n - 1)$;
2. Using [14, Algorithm 4], evaluate the τ-NAF of κ, $\kappa = \sum_i k_i \tau^i$;
3. Compute $\boldsymbol{Q} \leftarrow k\boldsymbol{P}$ as $\boldsymbol{Q} = \sum_i k_i \tau^i(\boldsymbol{P})$;
4. Return $\boldsymbol{Q}$.

Our randomization method exploits the structure of $\mathbb{Z}[\tau] \subseteq \mathrm{End}(E)$. Let $\rho \in \mathbb{Z}[\tau]$. If $x \equiv y \pmod{\rho(\tau^n - 1)}$ then x and y still act identically on the curve. Consequently, instead of reducing the multiplier modulo $\tau^n - 1$ (cf. Step 1 in the previous algorithm), we can reduce it modulo $\rho(\tau^n - 1)$ where ρ is a random element of $\mathbb{Z}[\tau]$. The length of the τ-NAF produced is approximately equal to $n + \log_2 \mathrm{N}(\rho)$, which penalizes the scalar multiplication by $\log_2 \mathrm{N}(\rho)$ additional steps. This enables to control very easily the trade-off between the running time and the expected security. Typically, for $n = 163$, we might impose that $\mathrm{N}(\rho) \approx 2^{40}$, which roughly produces τ-NAF of 200 digits (in $\{-1, 0, 1\}$) instead of 160 with the deterministic method. The detailed algorithm is presented below.

Algorithm 3 (Scalar Multiplication via Random Exponent Recoding for ABC Curves).

Input:	A point $\boldsymbol{P} = (x_1, y_1) \in E(\mathbb{F}_{2^n})$, an ABC curve. An integer k. A trade-off parameter l (typically, $l = 40$).
Output:	The point $\boldsymbol{Q} = k\boldsymbol{P}$.

1. Randomly choose an element $\rho \in \mathbb{Z}[\tau]$ with $\mathrm{N}(\rho) < 2^l$;
2. Compute, in $\mathbb{Z}[\tau]$, $\kappa' \leftarrow k \bmod \rho(\tau^n - 1)$;
3. Evaluate the τ-NAF of κ', $\kappa' = \sum_i \kappa'_i \tau^i$;
4. Compute $\boldsymbol{Q} \leftarrow \sum_i \kappa'_i \tau^i(\boldsymbol{P})$;
5. Return $\boldsymbol{Q}$.

An interesting feature of this algorithm is that no additional routine needs to be implemented. It only requires a slight modification of the deterministic

version. Furthermore, the random component ρ is spread over the full length of the multiplier. This may be better than simply adding to the multiplier a multiple of the order of the curve, as was suggested in [5, §. 5.1].

6 Conclusion

We proposed two new methods to blind the basepoint for an elliptic curve cryptosystem. These methods come from the idea of transposing the computation in another curve through a random morphism. In addition, we presented a new technique to randomize the encoding of the multiplier in the case of anomalous binary curves.

References

1. IEEE Std 1363-2000. *IEEE Standard Specifications for Public-Key Cryptography*. IEEE Computer Society, August 29, 2000.
2. D.V. Chudnovsky and G.V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Math.*, 7:385–434, 1986/7.
3. Henri Cohen. *A Course in Computational Algebraic Number Theory*. Number 138 in Graduate Texts in Mathematics. Springer-Verlag, 1993.
4. Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation using mixed coordinates. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65. Springer-Verlag, 1998.
5. Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES '99)*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer-Verlag, 1999.
6. Daniel M. Gordon. A survey on fast exponentiation methods. *Journal of Algorithms*, 27:129–146, 1998.
7. M. Anwar Hasan. Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 93–108. Springer-Verlag, 2000.
8. Neal Koblitz. CM-curves with good cryptographic protocols. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 279–287. Springer-Verlag, 1992.
9. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
10. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.
11. W. Meier and O. Staffelbach. Efficient multiplication on certain non-supersingular elliptic curves. In E.F. Brickell, editor, *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 333–344. Springer-Verlag, 1993.

12. Alfred J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
13. Atsuko Miyaji, Takatoshi Ono, and Henri Cohen. Efficient elliptic curve exponentiation. In Y. Han, T. Okamoto, and S. Qing, editors, *Information and Communications Security (ICICS '97)*, volume 1334 of *Lecture Notes in Computer Science*, pages 282–290. Springer-Verlag, 1997.
14. Jerome A. Solinas. An improved algorithm for arithmetic on a family of elliptic curves. In B. Kaliski, editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 357–371. Springer-Verlag, 1997.

A Mathematical Background

This appendix details the elliptic curve addition formulæ. It also reviews some well-known techniques for computing $\boldsymbol{Q} = k\boldsymbol{P}$ in an elliptic curve $E(\mathbb{K})$. An excellent survey by Gordon, including the most recent developments, can be found in [6].

In the sequel, we only consider the cases of a field $\mathbb{K}$ with $\mathrm{Char}\,\mathbb{K} \neq 2, 3$ and $\mathrm{Char}\,\mathbb{K} = 2$. In these cases, the general Weierstraß equation (cf. Eq. (1)) can be simplified considerably through an appropriate *admissible change of variables*. This is explicited in the next proposition.

Proposition 2 ([12, Theorem 2.2]). *The elliptic curves given by the Weierstraß equations*

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \text{ and}$$
$$E'_{/\mathbb{K}} : y^2 + a'_1xy + a'_3y = x^3 + a'_2x^2 + a'_4x + a'_6$$

are isomorphic over $\mathbb{K}$ if and only if there exists $u \in \mathbb{K}^*$ and $r, s, t \in \mathbb{K}$ such that the change of variables

$$(x, y) \leftarrow (u^2x + r, u^3y + u^2sx + t)$$

transforms equation E into equation E'. Such a transformation is referred to as an admissible change of variables. Furthermore,

$$\begin{cases} ua'_1 = a_1 + 2s\,, \\ u^2a'_2 = a_2 - sa_1 + 3r - s^2\,, \\ u^3a'_3 = a_3 + ra_1 + 2t\,, \\ u^4a'_4 = a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st\,, \\ u^6a'_6 = a_6 + ra_4 - ta_3 + r^2a_2 - rta_1 + r^3 - t^2\,. \end{cases}$$

A.1 Elliptic curves over a field $\mathbb{K}$ with $\mathrm{Char}\,\mathbb{K} \neq 2, 3$

When the characteristic of field $\mathbb{K}$ is different from $2, 3$, the Weierstraß equation of an elliptic curve can be simplified to:

$$E_{/\mathbb{K}} : y^2 = x^3 + ax + b \qquad (4a^3 + 27b^2 \neq 0)\ . \tag{10}$$

For any $\boldsymbol{P} \in E(\mathbb{K})$, we have $\boldsymbol{P} + \boldsymbol{O} = \boldsymbol{O} + \boldsymbol{P} = \boldsymbol{P}$. Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2) \in E(\mathbb{K})$. The inverse of $\boldsymbol{P}$ is $-\boldsymbol{P} = (x_1, -y_1)$. If $\boldsymbol{Q} = -\boldsymbol{P}$ then $\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{O}$; otherwise the sum $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ is given by

$$x_3 = \lambda^2 - x_1 - x_2\,, \quad y_3 = \lambda(x_1 - x_3) - y_1 \tag{11}$$

$$\text{with } \lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}\,, & \text{if } \boldsymbol{P} \neq \boldsymbol{Q}\,, \\ \dfrac{3x_1^2 + a}{2y_1}\,, & \text{if } \boldsymbol{P} = \boldsymbol{Q}\,. \end{cases}$$

To avoid the division in the computation of λ, one usually works in projective coordinates. There are basically two ways to project Eq. (10): (i) set $x = X/Z$ and $y = Y/Z$, that is, $(X : Y : Z)$ are the *homogeneous coordinates*; or (ii) set $x = X/Z^2$ and $y = Y/Z^3$, $(X : Y : Z)$ are then referred to as the *Jacobian coordinates*. Hence, to compute $\boldsymbol{Q} = k\boldsymbol{P}$ on an elliptic curve, one first represents point $\boldsymbol{P} = (x_1, y_1)$ as $(X_1 : Y_1 : Z_1)$, computes $(X_2 : Y_2 : Z_2) = k(X_1 : Y_1 : Z_1)$, and recovers $\boldsymbol{Q} = (x_2, y_2)$ from its projective form.

Homogeneous coordinates In homogeneous coordinates, $(X : Y : Z)$ and $(tX : tY : tZ)$ (with $t \in \mathbb{K}^*$) are two equivalent representations of a same point. The point at infinity $\boldsymbol{O}$ is represented by $(0 : 1 : 0)$; it is the only point with its Z-coordinate equal to 0. Putting $x = X/Z$ and $y = Y/Z$ in Eq. (12), the Weierstraß equation of an elliptic curve becomes

$$E_{/\mathbb{K}} : Y^2Z = X^3 + aXZ^2 + bZ^3 \ . \tag{12}$$

The formula to double a point $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ is $2\boldsymbol{P} = (X_3 : Y_3 : Z_3)$, where

$$X_3 = SH, \quad Y_3 = W(T - H) - 2M^2 \quad \text{and} \quad Z_3 = S^3 \tag{13}$$

with $W = 3X_1^2 + aZ_1^2$, $S = 2Y_1Z_1$, $M = Y_1S$, $T = 2X_1M$ and $H = W^2 - 2T$. This requires 12 multiplications. Notice that if $a = -3$ then $W = 3(X_1 - Z_1)(X_1 + Z_1)$; in that case, the number of multiplications decreases to 10. The sum $\boldsymbol{R} = (X_3 : Y_3 : Z_3)$ of two points $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$ (with $\boldsymbol{P} \neq \pm\boldsymbol{Q}$) is given by

$$X_3 = WX_3', \quad 2Y_3 = RV - MW^3 \quad \text{and} \quad Z_3 = Z'W^3 \tag{14}$$

with $U_1 = X_1Z_2$, $U_2 = X_2Z_1$, $S_1 = Y_1Z_2$, $S_2 = Y_2Z_1$, $T = U_1 + U_2$, $W = U_1 - U_2$, $M = S_1 + S_2$, $R = S_1 - S_2$, $Z' = Z_1Z_2$, $H = TW^2$, $X_3' = -H + Z'R^2$ and $V = H - 2X_3'$. The addition of two points can thus be done with only 14 multiplications. If one of the two points has its Z-coordinate equal to 1 then the number of multiplications decreases to 11.

Jacobian coordinates The use of Jacobian coordinates is suggested in the P1363 IEEE Standard [1] because it allows faster arithmetic [2]. In Jacobian

coordinates, also, the representation of points is not unique, $(X : Y : Z)$ and $(t^2X : t^3Y : tZ)$ (with $t \in \mathbb{K}^*$) are equivalent representations. The Weierstraß equation is given by

$$E_{/\mathbb{K}} : Y^2 = X^3 + aXZ^4 + bZ^6 \tag{15}$$

and the point at infinity is represented by $(1 : 1 : 0)$.

The double of point $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ is equal to $2\boldsymbol{P} = (X_3 : Y_3 : Z_3)$ where

$$X_3 = M^2 - 2S, \quad Y_3 = M(S - X_3) - T \quad \text{and} \quad Z_3 = 2Y_1Z_1 \tag{16}$$

with $M = 3X_1^2 + aZ_1^4$, $S = 4X_1Y_1^2$ and $T = 8Y_1^4$. So doubling a point requires 10 multiplications. Here too, we see that the value $a = -3$ enables to reduce the number of multiplications; in this case, it decreases to 8.

The sum $\boldsymbol{R} = (X_3 : Y_3 : Z_3)$ of points $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$ (with $\boldsymbol{P} \neq \pm\boldsymbol{Q}$) is given by

$$X_3 = R^2 - TW^2, \quad 2Y_3 = RV - MW^3 \quad \text{and} \quad Z_3 = Z_1Z_2W \tag{17}$$

with $U_1 = X_1Z_2^2, U_2 = X_2Z_1^2, S_1 = Y_1Z_2^3, S_2 = Y_2Z_1^3, T = U_1+U_2, W = U_1-U_2$, $M = S_1 + S_2$, $R = S_1 - S_2$ and $V = TW^2 - 2X_3$. An addition requires 16 multiplications. When one of the two points has its Z-coordinate equal to 1 then an addition requires only 11 multiplications. A slightly different (but equally efficient) formula for addition can be found in [13]. Using the same notations as above, the sum $\boldsymbol{R} = (X_3 : Y_3 : Z_3)$ is then given by

$$X_3 = R^2 - TW^2, \quad Y_3 = -RX_3 + (RU_1 - S_1)W^2 \quad \text{and} \quad Z_3 = Z_1Z_2W \ . \tag{17'}$$

Mixed coordinates In the general case, we have seen that Jacobian coordinates offer a faster doubling but a slower addition than homogeneous coordinates (see Table 2). Chudnovsky and Chudnovsky [2] proposed to internally represent a point $(X : Y : Z)$ in Jacobian coordinates as a 5-tuple (X, Y, Z, Z^2, Z^3). In *Chudnovsky Jacobian coordinates*, the addition formula for $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$, respectively represented as $(X_1, Y_1, Z_1, Z_1^2, Z_1^3)$ and $(X_2, Y_2, Z_2, Z_2^2, Z_2^3)$, remains the same as given by Eq. (17). The advantage is that the values of Z_1^2, Z_1^3, Z_2^2 and Z_2^3 being available, they do not have to be computed; only Z_3^2 and Z_3^3 have to be be computed to represent the result $\boldsymbol{R} = \boldsymbol{P} + \boldsymbol{Q} = (X_3 : Y_3 : Z_3)$ as the 5-tuple $(X_3, Y_3, Z_3, Z_3^2, Z_3^3)$. Therefore, Chudnovsky Jacobian coordinates require $(4 - 2) = 2$ multiplications less than ordinary Jacobian coordinates to add two points. On the other hand, the doubling is more expensive: it requires $(2-1) = 1$ multiplication more for computing $\boldsymbol{R} = 2\boldsymbol{P} = (X_3 : Y_3 : Z_3)$ since Z_1^2 has not to be computed (see Eq. (16)) but Z_3^2 and Z_3^3 have to.

The above strategy was optimized by Cohen, Miyaji and Ono [4] in order to provide the fastest known doubling algorithm on a general elliptic curve. With their coordinates, called *modified Jacobian coordinates*, a point $(X : Y : Z)$ is

Table 2. Number of multiplications in addition formulæ.

	Addition		Doubling	
	$Z_2 \neq 1$	$Z_2 = 1$	$a \neq -3$	$a = -3$
Homogeneous coord.	14	11	12	10
Jacobian coord.	16	11	10	8
Chudnovsky Jacobian coord.	14	11	11	9
Modified Jacobian coord.	19	14	8	8

internally represented as a 4-tuple (X, Y, Z, aZ^4). A point is doubled with only 8 multiplications *whatever* the value of parameter a. However, this fast doubling is done at the expense of a slower addition: 19 multiplications are required to add two points in the general case and 14 multiplications when one of the two points has its Z-coordinate equal to 1.

A.2 Elliptic curves over a field $\mathbb{K}$ with Char $\mathbb{K} = 2$

For fields of characteristic 2, the simplified Weierstraß equation depends on whether the curve is supersingular or not. For cryptographic applications, we are only interested in non-supersingular curves. In that case, it can be shown that an admissible change of variables yields the simplified Weierstraß equation

$$E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b \qquad (b \neq 0) \ . \tag{18}$$

$\boldsymbol{O}$ being the neutral element, we have $\boldsymbol{P}+\boldsymbol{O} = \boldsymbol{O}+\boldsymbol{P} = \boldsymbol{P}$ for any $\boldsymbol{P} \in E(\mathbb{K})$. Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2) \in E(\mathbb{K})$. The inverse of $\boldsymbol{P}$ is $-\boldsymbol{P} = (x_1, x_1 + y_1)$. If $\boldsymbol{Q} = -\boldsymbol{P}$ then $\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{O}$; otherwise the sum $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ is calculated as follows.

– If $\boldsymbol{P} \neq \boldsymbol{Q}$ then

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \,, \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \tag{19}$$

with $\lambda = \dfrac{y_1 + y_2}{x_1 + x_2}$.

– If $\boldsymbol{P} = \boldsymbol{Q}$ then

$$x_3 = \lambda^2 + \lambda + a \,, \quad y_3 = x_1^2 + (\lambda + 1)x_3 \tag{20}$$

with $\lambda = x_1 + \dfrac{y_1}{x_1}$.

An important subclass of elliptic curves has been introduced by Koblitz in [8]: the *Anomalous Binary Curves* (or ABC curves in short), sometimes also referred to as *Koblitz curves*. These are elliptic curves given by Eq. (18) with $b = 1$ and

$a \in \{0, 1\}$. For such curves, the Frobenius endomorphism, $\tau : (x, y) \mapsto (x^2, y^2)$, satisfies the characteristic equation

$$u^2 - (-1)^{1-a}\, u + 2 = 0 \ .$$

Koblitz suggests to speed the computation of $\boldsymbol{Q} = k\boldsymbol{P}$ by noticing that $2\boldsymbol{P} = (-1)^{1-a}\tau(\boldsymbol{P}) - \tau^2(\boldsymbol{P})$. He also suggests to write k as a Frobenius expansion since scalar multiplication by k is an endomorphism and $\mathbb{Z} \subseteq \mathbb{Z}[\tau] \subseteq \mathrm{End}(E)$. The ring $\mathbb{Z}[\tau]$ is an Euclidean domain with respect to the norm $\mathrm{N}(r + s\tau) = r^2 + (-1)^{1-a}\, rs + 2s^2$. Furthermore, as $\mathrm{N}(\tau) = 2$, every element $r + s\tau$ in $\mathbb{Z}[\tau]$ can be written as a *τ-adic non-adjacent form* (τ-NAF, in short), that is,

$$r + s\tau = \sum_i k_i\, \tau^i \quad \text{with} \begin{cases} k_i \in \{-1, 0, 1\} \\ k_i \cdot k_{i+1} = 0 \end{cases} \ . \tag{21}$$

As already remarked in [8], the drawback in this method is that the Frobenius expansion (21) is roughly twice longer than the usual balanced binary expansion and so, even if the evaluation of τ is very fast, it is not clear that the resulting method is faster. The drawback was loopholed in [11, 14] with the following observation. We obviously have $\tau^n = 1$ and thus $\boldsymbol{Q} = k'\boldsymbol{P}$ with $k' = k \bmod (\tau^n - 1)$. As $\mathrm{N}(\tau^n - 1) = \#E_a(\mathbb{F}_{2^n}) \approx 2^n$ by Hasse's Theorem, the τ-NAF expression of k', $k' = \sum_i k'_i\, \tau^i$, would have a length approximatively equal to that of the (usual) NAF expression of k. The non-adjacency property (i.e., $k'_i \cdot k'_{i+1} = 0$) implies that, on average, only one third of the digits are nonzero [6]. Together with the property that the evaluation of $\tau\boldsymbol{P}$ is very fast, this yields a very efficient algorithm for computing $\boldsymbol{Q} = k\boldsymbol{P}$.

Annexe B

Curriculum vitæ

Formation

Octobre 1997	Thèse de doctorat en Sciences Appliquées Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgique
Juin 1995	DEA en Mathématiques UCL (Louvain-la-Neuve, Belgique) et École Polytechnique (Palaiseau, France)
Janvier 1994	Ingénieur civil en Mathématiques Appliquées (niveau BAC+5) UCL (Louvain-la-Neuve, Belgique)

Expérience professionnelle

1999 –	Gemplus R&D, La Ciotat, France
1998 – 1999	Post-doctorat, Université de Tamkang, Taiwan, R.O.C.
1997 – 1998	Post-doctorat, Université catholique de Louvain (UCL)
1994 – 1997	Assistant, département de Mathématique, UCL
1994	Projet Européen *ACCOPI* (Access Control and COpyright Protection for Images)

Activités scientifiques

Comités de programme

- Comité directeur (steering committee) de RSA Conference
- Président de programme, RSA Conference 2003, Cryptographers' Track
- Participation à plusieurs comités de programme :
  - ASIACRYPT/AUSCRYPT : ASIACRYPT 2003
  - Cryptographic Hardware and Embedded Systems : CHES 2003
  - Public Key Cryptography : PKC 2003 et PKC 2004
  - Financial Cryptography : FC 2004
  - Information Security Conference : ISC 2003
  - Smart Card Research and Applications : CARDIS 2002
  - Information Security Solutions Europe : ISSE 2002
  - Workshop on Information Security Applications : WISA 2000, WISA 2001 et WISA 2002

Encadrement d'étudiants

- Mathieu Ciet, thèse de doctorat (UCL, Louvain-la-Neuve), 2001–2003
- Olivier Billet, DEA (U. Nice, Sophia Antipolis & Eurecom, Nice), 2002
- Aurore Gillet, diplôme d'ingénieur (UCL, Louvain-la-Neuve), 1997
- Alexis Bernard et Nicolas Degand, diplôme d'ingénieur (UCL, Louvain-la-Neuve), 1996

Domaines de recherche

- Cryptographie, sécurité de l'information, sécurité prouvée
- RSA, ECC, sécurité contre les attaques à canaux cachés et par fautes, implantations contraintes (carte à puce)
- Courbes elliptiques, algorithmique, arithmétique

Exposés invités

- Ruhr-Universität Bochum, «Elliptic Curve Cryptography and Side-Channel Analysis», Bochum, Allemagne, janvier 2003
- Azur'Crypt
  - «Sécurité des systèmes sur courbes elliptiques», Luminy, mars 2002
  - «Preventing Side-Channel Attacks in Elliptic Curve Cryptosystems : An Update», Toulon, février 2003
- AT&T Research Labs, «Security Paradoxes», Florham Park, NJ, États-Unis, août 1999
- NCKU Cryptology & Network Security Lab, «Introduction to Elliptic Curve Cryptography», Tainan, Taiwan, mai 1999
- École Normale Supérieure (ENS), «Analyse de la sécurité des cryptosystèmes du type RSA», Paris, octobre 1997

Divers

- Inventeur le plus prolifique à Gemplus (Gemplus' Management Team) en 2001 et 2002
- Nominé dans la catégorie «Meilleur software» (Cartes IT Security, Paris, novembre 2002)
- Meilleur papier (International Computer Symposium, Tainan, Taiwan, décembre 1998)
- Co-fondateur du *Groupe Crypto de l'UCL*, Louvain-la-Neuve, Belgique
- Membre de l'*International Association of Cryptologic Research* (IACR) depuis 1995

Références

Disponibles sur demande

Publications & communications

Livres et actes de conférences

1. Marc Joye, éditeur. *Topics in Cryptology – CT-RSA 2003*, volume 2612 de *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

Journaux

18. Benoît Chevallier-Mames, Mathieu Ciet, et Marc Joye. « Low-cost solutions for preventing simple side-channel analysis : Side-channel atomicity ». *IEEE Transactions on Computers*, À paraître.
17. Mathieu Ciet et Marc Joye. « Elliptic curve cryptosystems in the presence of permanent and transient faults ». *Designs, Codes and Cryptography*, À paraître.
16. Marc Joye. « Cryptanalysis of a pay-as-you-watch system ». *Information Processing Letters*, 88(3) : 119–120, 2003.
15. Marc Joye. « Elliptic curves and side-channel analysis ». *ST Journal of System Research*, 4(1) : 17–21, 2003.
14. Olivier Benoît et Marc Joye. « Protecting RSA against fault attacks ». *Smart Times*, 4(2), 2002.
13. Marc Joye. « Recovering lost efficiency of exponentiation algorithms on smart cards ». *Electronics Letters*, 38(19) : 1095–1097, 2002.
12. Marc Joye, Jean-Jacques Quisquater, et Tsuyoshi Takagi. « How to choose secret parameters for RSA and its extensions to elliptic curves ». *Designs, Codes and Cryptography*, 23(3) : 297–316, 2001.
11. Sung-Ming Yen et Marc Joye. « Checking before output may not be enough against fault-based cryptanalysis ». *IEEE Transactions on Computers*, 49(9) : 967–970, 2000.
10. Marc Joye et Sung-Ming Yen. « Optimal left-to-right binary signed-digit exponent recoding ». *IEEE Transactions on Computers*, 49(7) : 740–748, 2000.
9. Marc Joye, Arjen K. Lenstra, et Jean-Jacques Quisquater. « Chinese remaindering cryptosystems in the presence of faults ». *Journal of Cryptology*, 12(4) : 241–245, 1999.
8. Marc Joye et Sung-Ming Yen. « ID-based secret-key cryptography ». *ACM Operating Systems Review*, 32(4) : 33–39, 1998.
7. Sung-Ming Yen et Marc Joye. « Improved authenticated multiple-key agreement protocol ». *Electronics Letters*, 34(18) : 1738–1739, 1998.
6. Marc Joye et Jean-Jacques Quisquater. « Reducing the elliptic curve cryptosystem of Meyer-Müller to the cryptosystem of Rabin-Williams ». *Designs, Codes and Cryptography*, 14(1) : 53–56, 1998.
5. Marc Joye et Jean-Jacques Quisquater. Cryptanalysis of RSA-type cryptosystems : A visit. Dans R.N. Wright et P.G. Neumann, éditeurs, *Network Threats*, volume 38 de *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 21–31. American Mathematical Society, 1998.
4. Marc Joye et Jean-Jacques Quisquater. « Cryptosystem of Chua and Ling ». *Electronics Letters*, 33(23) : 1938, 1997.
3. Jean-François Dhem, Marc Joye, et Jean-Jacques Quisquater. « Normalisation in diminished-radix modulus transformation ». *Electronics Letters*, 33(23) : 1931, 1997.

2. Jean-Jacques Quisquater et Marc Joye. « Authentication of sequences with the SL_2 hash function : Application to video sequences ». *Journal of Computer Security*, 5(3) : 213–223, 1997.
1. Marc Joye et Jean-Jacques Quisquater. « Efficient computation of full Lucas sequences ». *Electronics Letters*, 32(6) : 537–538, 1996.

Conférences

43. Marc Joye et Pascal Paillier. « Constructive methods for the generation of prime numbers ». Dans *New European schemes for signatures, integrity, and encryption*, Lecture Notes in Computer Science. Springer-Verlag, À paraître.
42. Mathieu Ciet et Marc Joye. « (Virtually) free randomization techniques for elliptic curve cryptography ». Dans S. Qing, D. Gollmann, et J. Zhou, éditeurs, *Information and Communications Security (ICICS 2003)*, volume 2836 de *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, 2003.
41. Marc Joye et Pascal Paillier. « GCD-free algorithms for computing modular inverses ». Dans C.D. Walter, Ç.K. Koç, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 de *Lecture Notes in Computer Science*, pages 243–253. Springer-Verlag, 2003.
40. Benoît Chevallier-Mames, Marc Joye, et Pascal Paillier. « Faster double-size modular multiplication from Euclidean multiplier ». Dans C.D. Walter, Ç.K. Koç, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 de *Lecture Notes in Computer Science*, pages 214–227. Springer-Verlag, 2003.
39. Éric Brier et Marc Joye. « Fast point multiplication on elliptic curves through isogenies ». Dans M. Fossorier, T. Høholdt, et A. Poli, éditeurs, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 2643 de *Lecture Notes in Computer Science*, pages 43–50. Springer-Verlag, 2003.
38. Olivier Billet et Marc Joye. « The Jacobi model of an elliptic curve and side-channel analysis ». Dans M. Fossorier, T. Høholdt, et A. Poli, éditeurs, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 2643 de *Lecture Notes in Computer Science*, pages 34–42. Springer-Verlag, 2003.
37. Marc Joye et Karine Villegas. « A protected division algorithm ». Dans P. Honeyman, éditeur, *Smart Card Research and Advanced Applications (CARDIS '02)*, pages 69–74. Usenix Association, 2002.
36. Nathalie Feyt et Marc Joye. « A better use of smart cards in PKIs ». Dans *Gemplus Developer Conference*, Singapore, novembre 2002.
35. Nathalie Feyt, Marc Joye, David Naccache, et Pascal Paillier. « Off-line/on-line generation of RSA keys with smart cards ». Dans S.-P. Shieh, éditeur, *2nd International Workshop for Asian Public Key Infrastructures (IWAP 2002)*, pages 153–158, Taipei, Taiwan, octobre 2002.
34. Marc Joye et Pascal Paillier. « How to use RSA ; or how to improve the efficiency of RSA without loosing its security ». Dans *ISSE 2002*, Paris, France, octobre 2002.
33. Marc Joye et Sung-Ming Yen. « The Montgomery powering ladder ». Dans B.S. Kaliski Jr, Ç.K. Koç, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 de *Lecture Notes in Computer Science*, pages 291–302. Springer-Verlag, 2003.

32. Jean-Sébastien Coron, Marc Joye, David Naccache, et Pascal Paillier. « Universal padding schemes for RSA ». Dans M. Yung, éditeur, *Advances in Cryptology – CRYPTO 2002*, volume 2442 de *Lecture Notes in Computer Science*, pages 226–241. Springer-Verlag, 2002.
31. Marc Joye et Sung-Ming Yen. « New minimal modified radix-r representation with applications to smart cards ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 375–384. Springer-Verlag, 2002.
30. Marc Joye et Sung-Ming Yen. « One-way cross-trees and their applications ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 346–356. Springer-Verlag, 2002.
29. Éric Brier et Marc Joye. « Weierstraß elliptic curves and side-channel attacks ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 335–345. Springer-Verlag, 2002.
28. Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, et Christophe Tymen. « Optimal chosen-ciphertext secure encryption of arbitrary-length messages ». Dans D. Naccache et P. Paillier, éditeurs, *Public Key Cryptography*, volume 2274 de *Lecture Notes in Computer Science*, pages 17–33. Springer-Verlag, 2002.
27. Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, et Christophe Tymen. « GEM : A generic chosen-ciphertext secure encryption method ». Dans B. Preneel, éditeur, *Topics in Cryptology – CT-RSA 2002*, volume 2271 de *Lecture Notes in Computer Science*, pages 263–276. Springer-Verlag, 2002.
26. Marc Joye, Jean-Jacques Quisquater, Sung-Ming Yen, et Moti Yung. « Observability Analysis : Detecting when improved cryptosystems fail ». Dans B. Preneel, éditeur, *Topics in Cryptology – CT-RSA 2002*, volume 2271 de *Lecture Notes in Computer Science*, pages 17–29. Springer-Verlag, 2002.
25. Seungjoo Kim, Junghee Cheon, Marc Joye, Seongan Lim, Masahiro Mambo, Dongho Won, et Yuliang Zheng. « Strong adaptive chosen-ciphertext attack with memory dump (or : The importance of the order of decryption and validation) ». Dans B. Honary, éditeur, *Cryptography and Coding*, volume 2260 de *Lecture Notes in Computer Science*, pages 114–127. Springer-Verlag, 2001.
24. Marc Joye et Jean-Jacques Quisquater. « On Rabin-type signatures ». Dans B. Honary, éditeur, *Cryptography and Coding*, volume 2260 de *Lecture Notes in Computer Science*, pages 99–113. Springer-Verlag, 2001.
23. Marc Joye, Pascal Paillier, et Sung-Ming Yen. « Secure evaluation of modular functions ». Dans R.J. Hwang et C.K. Wu, éditeurs, *2001 International Workshop on Cryptology and Network Security*, pages 227–229, Taipei, Taiwan, septembre 2001.
22. Marc Joye et Pascal Paillier. « Constructive methods for the generation of prime numbers ». Dans S. Murphy, éditeur, *2nd NESSIE Workshop*, Egham, UK, septembre 2001.
21. Christophe Clavier et Marc Joye. « Universal exponentiation algorithm : A first step towards provable SPA-resistance ». Dans Ç.K. Koç, D. Naccache, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 de *Lecture Notes in Computer Science*, pages 300–308. Springer-Verlag, 2001.

20. Marc Joye et Christophe Tymen. « Protections against differential analysis for elliptic curve cryptography : An algebraic approach ». Dans Ç.K. Koç, D. Naccache, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 de *Lecture Notes in Computer Science*, pages 377–390. Springer-Verlag, 2001.
19. Marc Joye et Jean-Jacques Quisquater. « Hessian elliptic curves and side-channel attacks ». Dans Ç.K. Koç, D. Naccache, et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 de *Lecture Notes in Computer Science*, pages 402–410. Springer-Verlag, 2001.
18. Marc Joye, Jean-Jacques Quisquater, et Moti Yung. « On the power of misbehaving adversaries and security analysis of the original EPOC ». Dans D. Naccache, éditeur, *Topics in Cryptology – CT-RSA 2001*, volume 2020 de *Lecture Notes in Computer Science*, pages 208–222. Springer-Verlag, 2001.
17. Marc Joye et Christophe Tymen. « Compact encoding of non-adjacent forms with applications to elliptic curve cryptography ». Dans K. Kim, éditeur, *Public Key Cryptography*, volume 1992 de *Lecture Notes in Computer Science*, pages 353–364. Springer-Verlag, 2001.
16. Marc Joye, Pascal Paillier, et Serge Vaudenay. « Efficient generation of prime numbers ». Dans Ç.K. Koç et C. Paar, éditeurs, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 de *Lecture Notes in Computer Science*, pages 340–354. Springer-Verlag, 2000.
15. Giuseppe Ateniese, Jan Camenisch, Marc Joye, et Gene Tsudik. « A practical and provably secure coalition-resistant group signature scheme ». Dans M. Bellare, éditeur, *Advances in Cryptology – CRYPTO 2000*, volume 1880 de *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, 2000.
14. Jean-Sébastien Coron, Marc Joye, David Naccache, et Pascal Paillier. « New attacks on PKCS #1 v1.5 encryption ». Dans B. Preneel, éditeur, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 de *Lecture Notes in Computer Science*, pages 369–381. Springer-Verlag, 2000.
13. Marc Joye, Narn-Yih Lee, et Tzonelih Hwang. « On the security of the Lee-Chang group signature scheme and its derivatives ». Dans M. Mambo et Y. Zheng, éditeurs, *Information Security*, volume 1729 de *Lecture Notes in Computer Science*, pages 47–51. Springer-Verlag, 1999.
12. Marc Joye, Seungjoo Kim, et Narn-Yih Lee. « Cryptanalysis of two group signature schemes ». Dans M. Mambo et Y. Zheng, éditeurs, *Information Security*, volume 1729 de *Lecture Notes in Computer Science*, pages 271–275. Springer-Verlag, 1999.
11. Giuseppe Ateniese, Marc Joye, et Gene Tsudik. « On the difficulty of coalition-resistance in group signature schemes ». Dans G. Persiano, éditeur, *2nd Workshop on Security in Communication Networks (SCN '99)*, Almafi, Italy, septembre 1999.
10. Marc Joye, Jean-Jacques Quisquater, Sung-Ming Yen, et Moti Yung. « Security paradoxes : How improving a cryptosystem may weaken it ». Dans *Ninth National Conference on Information Security*, pages 27–32, Taichung, Taiwan, mai 1999.
9. Marc Joye et Richard Pinch. « Cheating in split-knowledge RSA parameter generation ». Dans D. Augot et C. Carlet, éditeurs, *Workshop on Coding and Cryptography*, pages 157–163, Paris, France, janvier 1999.

8. Marc Joye et Sung-Ming Yen. « Generation/Release of secrets using one-way cross-trees ». Dans T.L. Hwang et A.K. Lenstra, éditeurs, *Workshop on Cryptology and Information Security*, 1998 International Computer Symposium, pages 23–28, Tainan, Taiwan, décembre 1998.
7. Marc Joye, Jean-Jacques Quisquater, et Tsuyoshi Takagi. « Are strong primes really stronger ? ». Dans D.J. Guan, éditeur, *Eight National Conference on Information Security*, pages 355–367, Kaoshung, Taiwan, mai 1998.
6. Marc Joye, Jean-Jacques Quisquater, Feng Bao, et Robert H. Deng. « RSA-type signatures in the presence of transient faults ». Dans M. Darnell, éditeur, *Cryptography and Coding*, volume 1355 de *Lecture Notes in Computer Science*, pages 155–160. Springer-Verlag, 1997.
5. Daniel Bleichenbacher, Marc Joye, et Jean-Jacques Quisquater. « A new and optimal chosen-message attack on RSA-type cryptosystems ». Dans Y. Han, T. Okamoto, et S. Qing, éditeurs, *Information and Communications Security*, volume 1334 de *Lecture Notes in Computer Science*, pages 302–313. Springer-Verlag, 1997.
4. Aurore Gillet, Marc Joye, et Jean-Jacques Quisquater. « Cautionary note for protocols designers : Security proof is not enough ». Dans H. Orman et C. Meadows, éditeurs, *DIMACS Workshop on Design and Formal Verification of Security Protocols*, New-York, USA, septembre 1997. Disponible sur CDROM.
3. Marc Joye et Jean-Jacques Quisquater. « On the importance of securing your bins : The garbage-man-in-the-middle attack ». Dans T. Matsumoto, éditeur, *4th ACM Conference on Computer and Communications Security*, pages 135–141. ACM Press, 1997.
2. Marc Joye et Jean-Jacques Quisquater. « Protocol failures for RSA-type cryptosystems using Lucas sequences and elliptic curves ». Dans M. Lomas, éditeur, *Security Protocols*, volume 1189 de *Lecture Notes in Computer Science*, pages 93–100. Springer-Verlag, 1997.
1. Jean-Jacques Quisquater, Benoît Macq, Marc Joye, Nicolas Degand, et Alexis Bernard. « Practical solution to authentication of images with a secure camera ». Dans I.K. Sethi et R.C. Jain, éditeurs, *Storage and Retrieval for Image and Video Databases V*, volume 3022, pages 290–297. SPIE, 1997.

Rapports techniques

6. Marc Joye et Sung-Ming Yen. « New applications of digital signatures ». Final Report (NSC88-2213-E-032-003 and NSC88-2811-E-032-0001), National Science Council of the Republic of China, Taiwan, juillet 1999.
5. Marc Joye et Sung-Ming Yen. « Overall study and countermeasure developments of the hardware-oriented cryptanalysis ». Final Report (NSC87-2213-E-032-012 and NSC87-2811-E-032-0001), National Science Council of the Republic of China, Taiwan, août 1998.
4. Sung-Ming Yen, Marc Joye, et Pao-Yu Ku. « An improved cryptographic checksum algorithm based on stream ciphers ». Rapport Technique TR-98-1, Tamkang LCIS, Tamsui, février 1998.
3. Marc Joye et Jean-Jacques Quisquater. « Faulty RSA encryption ». Rapport Technique CG-1997/8, UCL Crypto Group, Louvain-la-Neuve, juillet 1997.

2. Marc Joye, François Koeune, et Jean-Jacques Quisquater. « Takagi/Naito's algorithm revisited ». Rapport Technique CG-1997/3, UCL Crypto Group, Louvain-la-Neuve, mars 1997.
1. Jean-Marc Boucqueau, Jean-François Delaigle, Jean-François Dhem, Marc Joye, François Koeune, Henri Massias, Patrick Mestré, et Jean-Jacques Quisquater. « Comment jouer à pile ou face sur Internet sans tricher ». Rapport Technique CG-1997/2, UCL Crypto Group, Louvain-la-Neuve, mars 1997.

Thèses

3. Marc Joye. « *Security analysis of RSA-type cryptosystems* ». Thèse de doctorat en Sciences Appliquées, Université catholique de Louvain, Louvain-la-Neuve, novembre 1997.
2. Marc Joye. « Introduction élémentaire à la théorie des courbes elliptiques ». Mémoire de DEA en mathématiques, Université catholique de Louvain, Louvain-la-Neuve, juin 1995.
1. Marc Joye. « Arithmétique algorithmique : Application au crypto-système à clé publique RSA ». Mémoire de fin d'études d'Ingénieur Civil en Mathématiques Appliquées, Université catholique de Louvain, Louvain-la-Neuve, janvier 1994.