Published in H. Jin and G. Heileman, Eds, *9th ACM Workshop in Digital Rights Management (ACM-DRM 2009)*, pp. 11–16, ACM Press, 2009.

A Simple Construction for Public-Key Encryption with Revocable Anonymity: The Honest-Sender Case

(Extended Abstract)

Davide Alessio
Université de Rennes 1, IRMAR &
Thomson R&D, Security Competence Center
Cesson-Sévigné, France
davide.alessio@thomson.net

Marc Joye
Thomson R&D, Security Competence Center
Cesson-Sévigné, France
marc.joye@thomson.net

ABSTRACT

This paper presents a simple and generic transformation that adds traceability to an anonymous encryption scheme. We focus on the case of honest senders, which finds applications in many real-life scenarios. Advantageously, our transformation can be applied to already deployed public-key infrastructures. Two concrete implementations are provided.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; E.3 [**Data**]: Data Encryption—*Public key cryptosystems*

General Terms

Security, Design

Keywords

Public-key encryption, key-privacy, anonymity

1. INTRODUCTION

In numerous scenarios, the recipient's identity in a transmission needs to be kept private. This allows users to maintain some privacy. Protecting communication content may be not enough, as already observed in a couple of papers (e.g., [1, 2, 13]). For example, by analyzing the traffic between an antenna and a mobile device, one can recover some information about [at least] user's position and some details about the use of her mobile device. This information leaks easily during all day: it is a common habit, indeed, to use a mobile phone every day and to keep it (almost) always switched on. A similar problem exists in the context of electronic commerce. If no anonymity is provided, users' preferences can be known. The knowledge of this information enables profiling users and sending them targeted advertisements or selling profiles to other commercial entities. It should be conceivable that buyers can make their choices and shop on Internet without risking to be profiled. On the other hand, it is understandable that, in some situations, there may be a need to be able to revoke the anonymity; for example in the case of a legal dispute — while keeping the content private. This paper provides such a solution, offering a trade-off between the needs of the different involved parties.

We propose constructions for encryption schemes with revocable anonymity. As a result, we get schemes keeping private the recipient's identity in a transmission, but offering to some trusted party the possibility to discover it.

Back to the wireless communication case, we can imagine antennas broadcasting messages without leaking receivers' identifiers to mobile devices and in case of network misuse, find out the user identifier and revoke her. The primitive can be deployed on a shared network (or storage) system where all data are encrypted. If someone overuses the resources quota, the system administrator can easily individuate that user and take appropriate actions (as a warning, reduce/revoke rights, ...).

Related work.

Formal notions of anonymity in the public-key setting appear in [2]. There should be no way for an adversary to distinguish between a message sent to a given recipient and one addressed to a random one. In [12], Pointcheval and Izabachène analyze different anonymity levels for identity-based encryption schemes ([14]).

Concurrent to our work, Kiayias *et al.* introduce and model in [13] the concept of *group encryption.* This is the analogue for encryption of group signatures [7]. Group encryption allows one to conceal the identity of the recipient of a given ciphertext among a set of legitimate receivers. However, in case of misuse, some authority (the group manager) is capable of recovering the recipient's identity. Furthermore, in addition to security and privacy properties, group encryption offers *verifiability*: a sender can convince a verifier that the formed ciphertext can be decrypted by a group member.

Our contribution.

Key privacy in public-key encryption assumes a "homogeneous" environment. Indeed, if users make use of different cryptosystems or of the same cryptosystem but with keys of different lengths, anonymity is likely to be lost. The notion of anonymity is therefore is restricted to users sharing the same cryptosystem and common parameters. This implicitly defines a group.

In this paper, we relax the requirements for group encryption. In the particular context of media broadcasting or wireless communications, we face a different situation where the sender (the broadcaster or the wireless emitter) can be trusted. This relaxation is justified by the fact that, in practical uses of the infrastructure, the sender has no interest in cheating because of business and reputation aspects. Moreover, it is very unlikely that an attacker can impersonate the sender, due to the particular material infrastructure needed (expensive, powerful, ...). Such an attacker should, indeed, mute the licit signals and substitute them with illicit ones, keeping all existing communications alive and faking the attacked ones. For similar reasons, we can also suppose that there are no collusions between senders and recipients. We refer to this setting as the *honest-sender case*.

The rest of this paper is organized as follows. In the next section, we introduce some background on public-key encryption. In Section 3, we define the notion of revocable anonymity. Section 4 is the core of our paper. We describe a simple and generic transformation to get a public-key encryption scheme with revocable anonymity. Finally, we present applications of our transformation in Section 5.

Acknowledgments.
We are grateful to the anonymous referees and to Alain Durand, Mohamed Karroumi and Nicolas Prigent for useful comments.

2. PRELIMINARIES

In this section, we review classical notions for public-key encryption, dealing both with data-privacy and key-privacy. We also introduce some useful notation.

Public-key encryption scheme.

In order to better capture the property that users may share some common parameters in a homogeneous environment, the key generation algorithm is divided in two sub-algorithms: the *common-key generation* algorithm and the *key generation* algorithm.

Following the syntax of [2], we define a *public-key encryption scheme* is a tuple of four algorithms $(\mathsf{CKeyGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

Common-key generation The common-key generation algorithm $\mathsf{CKeyGen}$ takes on input some security parameter λ and outputs some common key $I \overset{R}{\leftarrow} \mathsf{CKeyGen}(\lambda)$.

Key generation The key generation algorithm KeyGen is a randomized algorithm that takes on input I and returns a matching pair of public key and secret key for some user: $(\mathsf{upk}, \mathsf{usk}) \overset{R}{\leftarrow} \mathsf{KeyGen}(I)$.

Encryption Let $\mathcal{M}$ denote the message space. The encryption algorithm Enc is a randomized algorithm that takes on input a public key upk and a plaintext $m \in \mathcal{M}$, and returns a ciphertext c. We write $c \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m)$.

Decryption The decryption algorithm Dec takes on input secret key usk (matching upk) and ciphertext c and returns the corresponding plaintext m or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $m \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if c is a valid ciphertext and $\perp \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if it is not.

We require that $\mathsf{Dec}_{\mathsf{usk}}(\mathsf{Enc}_{\mathsf{upk}}(m)) = m$ for any message $m \in \mathcal{M}$.

Indistinguishability of encryptions.

The notion of *indistinguishability of encryptions* [11] captures a strong notion of [data]-privacy: The adversary should not learn any information whatsoever about a plaintext given its encryption beyond the length of the plaintext.

We view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. This corresponds to adversary $\mathcal{A}$ running in two stages. In the "find" stage, algorithm $\mathcal{A}_1$ takes on input a public key upk and outputs two equal-size messages m_0 and $m_1 \in \mathcal{M}$ and some state information s. In the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext c which is the encryption of m_b under upk and where b is chosen at random in $\{0,1\}$. The goal of $\mathcal{A}_2$ is to recover the value of b from s and c.

A public-key encryption scheme is said *semantically secure* (or *indistinguishable*) if

$$\Pr\left[\begin{array}{l} I \overset{R}{\leftarrow} \mathsf{CKeyGen}(\lambda), (\mathsf{upk},\mathsf{usk}) \overset{R}{\leftarrow} \mathsf{KeyGen}(I), \\ (m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{upk}), b \overset{R}{\leftarrow} \{0,1\}, c \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m_b) \end{array} : \mathcal{A}_2(s,c) = b\right] - \frac{1}{2}$$

is negligible in the security parameter for any polynomial-time adversary $\mathcal{A}$; the probability is taken over the random coins of the experiment according to the distribution induced by $\mathsf{CKeyGen}$ and KeyGen and over the random coins of the adversary.

As we are in the public-key setting, it is worth noting that adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is given the public key upk and so can encrypt any message of its choice. In other words, the adversary can mount chosen-plaintext attacks (CPA). Hence, we write *IE-CPA* the security notion achieved by a semantically secure encryption scheme.[1]

A stronger scenario is to give the adversary an adaptive access to a decryption oracle. The previous definition readily extends to this model. Adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is allowed to submit any ciphertext of its choice and receives the corresponding plaintext (or $\perp$); the sole exception is that $\mathcal{A}_2$ may not query the decryption oracle on challenge ciphertext c. Likewise, we write *IE-CCA* the corresponding security notion.

Indistinguishability of keys.

Analogously, the notion of *indistinguishability of keys* captures a strong requirement about key privacy: The adversary should not be able to link whatsoever a ciphertext with its underlying encryption key.

As before, we view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. In the "find" stage, algorithm $\mathcal{A}_1$ takes on input two public keys upk_0 and upk_1 and outputs

[1]We deviate from the usual notation of IND-CPA to emphasize the fact that indistinguishability is about encryptions.

a message m and some state information s. Then in the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext c which is the encryption of m under upk_b where b is chosen at random in $\{0,1\}$. The goal of $\mathcal{A}_2$ is to recover the value of b from s and c.

More formally, a public-key encryption scheme is said *anonymous* (or *key-private*) if

$$\Pr\left[\begin{array}{l} I \overset{R}{\leftarrow} \mathsf{CKeyGen}(\lambda), (\mathsf{upk}_0, \mathsf{usk}_0) \overset{R}{\leftarrow} \mathsf{KeyGen}(I), \\ (\mathsf{upk}_1, \mathsf{usk}_1) \overset{R}{\leftarrow} \mathsf{KeyGen}(I), (m, s) \leftarrow \mathcal{A}_1(\mathsf{upk}_0, \mathsf{upk}_1), : \\ b \overset{R}{\leftarrow} \{0,1\}, c \leftarrow \mathsf{Enc}_{\mathsf{upk}_b}(m) \end{array} \quad \mathcal{A}_2(s, c) = b\right] - \frac{1}{2}$$

is negligible in the security parameter for any polynomial-time adversary $\mathcal{A}$; the probability is taken over the random coins of the experiment according to the distribution induced by $\mathsf{CKeyGen}$ and KeyGen and over the random coins of the adversary.

This definition of anonymity gives rise to the security notion of *IK-CPA* or *indistinguishability of keys under chosen-plaintext attacks.* If the adversary is given adaptive access to a decryption oracle, the corresponding security notion is *IK-CCA* or *indistinguishability of keys under chosen-ciphertext attacks.*

Of course, the goals of data-privacy and key-privacy can be combined to define extended security notions. A public-key encryption scheme achieves *IND-CPA security* if it is both IE-CPA and IK-CPA. Likewise, a public-key encryption scheme achieves *IND-CCA security* if it is both IE-CCA and IK-CCA.

3. REVOCABLE ANONYMITY

As exemplified in the introduction, there are several use cases where the sender is trustful and has no incentive to cheat. However, while it may be useful to keep the identity of the receiver private, it may also be useful to have some means to discover the identity of a receiver in case of misuse or dispute. This section formally defines the notion of revocable anonymity.

3.1 Formal definition

We augment the definition of public-key encryption scheme so that anonymity can be revoked. The formalization shares many parts with that of a (regular) public-key encryption scheme. The main differences are *(i)* the generation of a pair of keys for tracing purposes in the common-key generation and *(ii)* a tracing algorithm for recovering the intended recipient of a ciphertext.

More formally, a *public-key encryption scheme with revocable anonymity* is a tuple of five algorithms ($\mathsf{CKeyGen}$, KeyGen, Enc, Dec, Trace):

Common-key generation The key generation algorithm $\mathsf{CKeyGen}$ takes on input some security parameter λ and

1. generates the opener's secret and public keys osk and opk;
2. outputs some common key I: $I \overset{R}{\leftarrow} \mathsf{CKeyGen}(\lambda)$.

Here, element I may include a copy of public key opk.

Key generation The key generation algorithm KeyGen is a randomized algorithm that takes on input I and returns a matching pair of public key and secret key for some user: $(\mathsf{upk}, \mathsf{usk}) \overset{R}{\leftarrow} \mathsf{KeyGen}(I)$.

Encryption Let $\mathcal{M}$ denote the message space. The encryption algorithm Enc is a randomized algorithm that takes on input a public key upk and a plaintext $m \in \mathcal{M}$ and returns a ciphertext c. We write $c \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m)$.

Decryption The decryption algorithm Dec takes on input secret key usk (matching upk) and ciphertext c and returns the corresponding plaintext m or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $m \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if c is a valid ciphertext and $\perp \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if it is not.

Tracing The tracing algorithm Trace takes on input secret opening key osk and ciphertext c and returns the corresponding user's public key upk or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $\mathsf{upk} \leftarrow \mathsf{Trace}_{\mathsf{osk}}(c)$ if c is a valid ciphertext and $\perp \leftarrow \mathsf{Trace}_{\mathsf{osk}}(c)$ if is not.

We require for a public-key encryption scheme with revocable anonymity the two following properties:

Correctness For any message $m \in \mathcal{M}$ and for any pair of matching public key/secret key returned by the key generation algorithm, $(\mathsf{upk}, \mathsf{usk}) \overset{R}{\leftarrow} \mathsf{KeyGen}(I)$, one has $\mathsf{Dec}_{\mathsf{usk}}(\mathsf{Enc}_{\mathsf{upk}}(m)) = m$; and

Traceability For any message $m \in \mathcal{M}$ and for any pair of matching public key/secret key returned by the key generation algorithm, $(\mathsf{upk}, \mathsf{usk}) \overset{R}{\leftarrow} \mathsf{KeyGen}(I)$, one has $\mathsf{Trace}_{\mathsf{osk}}(\mathsf{Enc}_{\mathsf{upk}}(m)) = \mathsf{upk}$.

Given a [well formed] ciphertext, the first property (*correctness*) ensures that the intended recipient will always recover the corresponding plaintext while the second property (*traceability*) ensures that the tracing authority will always discover the recipient, if needed.

3.2 Non-malleability

In this section we add a further requirement to enforce the *traceability* property and discuss the notion of *non-malleability.*

A limited solution.

A simple way to get revocable anonymity is to *(i)* encrypt the message as usual with a key-private public-key encryption scheme, *(ii)* encrypt the recipient's identity under a trusted public encryption-key, and *(iii)* define the resulting ciphertext as the concatenation of the two encryptions and make it available to the recipient.

It is easily verified that the above scheme meets the *correctness* and *traceability* properties. But it also suffers from limitations, even in the honest-sender-case. The *traceability* property resides only in the presence of the encryption of the recipient's identity. If, for various reasons, this encryption is modified or suppressed, the trusted authority would no longer be able to discover the recipient's identity. On the other hand, the recipient might still be able to decrypt the ciphertext and recover the corresponding plaintext message.

REMARK. We note that signing the ciphertext (or similar techniques) does not solve the problem — or does, in addition, require compliant decrypting hardware to check the ciphertext validity and return the corresponding plaintext message only if the test is successful.

Non-malleability.

It is useful to introduce some notation We extend the decryption algorithm to vectors of ciphertexts (denoted in boldface). If $\boldsymbol{c} = (c_1, \ldots, c_n)$, $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$ stands for $(\mathsf{Dec}_{\mathsf{sk}}(c_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(c_n))$.

Basically, with non-malleability [9, 3], the goal of the adversary is, given a ciphertext $\hat{c}$, to output a vector $\boldsymbol{c}$ of ciphertexts whose decryption, $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$, is "meaningfully" related to $\mathsf{Dec}_{\mathsf{sk}}(\hat{c})$. We write $\mathcal{R}(\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c}), \mathsf{Dec}_{\mathsf{sk}}(\hat{c})) = 1$ for some relation $\mathcal{R}$. Suppose now that the ciphertext output by the above scheme (limited solution) is non-malleable. As a result, if the tracing authority is not able to trace the ciphertext then its intended recipient would not be able to decrypt it. This prevents ciphertext modifications.

The requirement of non-malleability can be captured as the advantage of an adversary $\mathcal{A}$ running some experiment [3] (see also [4]).

Define experiment Exp_b by

$$\begin{cases} I \xleftarrow{R} \mathsf{CKeyGen}(\lambda), (\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{KeyGen}(I), \\ (\mathcal{M}, s) \leftarrow \mathcal{A}_1(\mathsf{upk}, \mathsf{opk}), \\ m_0, m_1 \xleftarrow{R} \mathcal{M}, \hat{c} \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m_1), \\ (R, \boldsymbol{c}) \leftarrow \mathcal{A}_2(\mathcal{M}, s, \hat{c}) \end{cases} : \quad R(\mathsf{Dec}_{\mathsf{usk}}(\boldsymbol{c}), m_b)) = 1 \ .$$

Ciphertext vector $\boldsymbol{c}$ returned by the adversary is supposed to be valid and not containing $\hat{c}$ (i.e., $\perp \notin \mathsf{Dec}_{\mathsf{usk}}(\boldsymbol{c})$ and $\hat{c} \notin \boldsymbol{c}$).

This leads to the property of non-malleability, adapted to our purposes:

Non-malleability For any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage

$$\Pr[\mathsf{Exp}_1(\lambda) = 1] - \Pr[\mathsf{Exp}_0(\lambda) = 1]$$

has to be negligible in security parameter λ whenever $\mathsf{Trace}_{\mathsf{osk}}(\boldsymbol{c}) \neq \mathsf{upk}$.

In the previous definition, the inequality $\mathsf{Trace}_{\mathsf{osk}}(\boldsymbol{c}) \neq \mathsf{upk}$ should not be understood in the strict sense as a mere application of the trace algorithm to the components of $\boldsymbol{c}$. It should be interpreted as "the tracing manager (owning secret opening key osk) cannot recover upk from $\boldsymbol{c}$."

4. TRANSFORMATION

We are now ready to present our transformation. It takes on input a key-private public-key encryption scheme, say $\mathcal{S}_{\mathrm{Msg}}$, and outputs a key-private public encryption scheme with revocable anonymity. Input scheme $\mathcal{S}_{\mathrm{Msg}} = (\mathsf{CKeyGen}, \mathsf{KeyGen}, \mathsf{MsgEnc}, \mathsf{MsgDec})$ is supposed to meet [at least] the IK-CPA security notion.

The transformation requires the presence of a trusted party, called *tracing authority*. The tracing authority owns a pair of keys $(\mathsf{opk}, \mathsf{osk})$ registered for public-key encryption scheme $\mathcal{S}_{\mathrm{ID}} = (\mathsf{IDCKeyGen}, \mathsf{IDKeyGen}, \mathsf{IDEnc}, \mathsf{IDDec})$, meeting [at least] the IE-CPA security notion.

The output scheme $\mathcal{S}'$, after the transformation, is detailed into the following five algorithms: $\mathcal{S}' = (\mathsf{RevCKeyGen}, \mathsf{RevKeyGen}, \mathsf{RevEnc}, \mathsf{RevDec}, \mathsf{Trace})$. Specifically, we have:

Common-key generation This algorithm takes on input a security parameter λ and outputs common parameter: $I \xleftarrow{R} \mathsf{RevCKeyGen}(\lambda) = \mathsf{CKeyGen}(\lambda)$ — so for the first algorithm the new scheme borrows the first algorithm of the original $\mathcal{S}_{\mathrm{Msg}}$. As mentioned before, I may include a copy of public key opk.

This algorithm also selects a non-malleable symmetric encryption scheme $\mathcal{E}$ and a second-preimage resistant key-derivation function KDF, whose descriptions are made public.

Key generation This randomized algorithm is run by some user. Given as an input common parameter I, it outputs a pair of matching public encryption key and secret decryption key, $(\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{RevKeyGen}(I) = \mathsf{KeyGen}(I)$.

Public-key usk is registered and certified by some authority (typically, the tracing authority).

Encryption This randomized algorithm is used to encrypt messages. It takes on input public keys opk and upk (i.e., tracing authority's public key and recipient's public-key), message m to be encrypted, and outputs:

$$\begin{aligned} c &\leftarrow \mathsf{RevEnc}_{\mathsf{opk}}(\mathsf{upk}, m) \\ &= (\mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk}), \mathsf{MsgEnc}_{\mathsf{upk}}(\mathcal{E}_k(m)) \end{aligned}$$

with $k = \mathrm{KDF}(\mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk}))$ is a key derived from the first part with key derivation function KDF.

Decryption This algorithm is run by the intended recipient, owning secret key usk. Taking on input ciphertext (c_1, c_2) and usk, this algorithm computes $k' = \mathrm{KDF}(c_1)$ and next $\mathcal{E}_{k'}^{-1}(\mathsf{MsgDec}_{\mathsf{usk}}(c_2))$, which yields corresponding plaintext message m (or $\perp$).

Tracing This algorithm is run by the tracing authority. On input the first part c_1 of ciphertext $c = (c_1, c_2)$ and secret opening key osk, it returns the public key corresponding to the intended recipient (or $\perp$), $\mathsf{upk} \leftarrow \mathsf{Trace}_{\mathsf{osk}}(c_1) = \mathsf{IDDec}_{\mathsf{osk}}(c_1)$.

Discussion.

We remind that we are in the honest-sender setting. In particular, the sender does not encapsulate fake or wrong identities.

The building blocks needed in the transformation require to satisfy some properties. The transformation splits the RevEnc algorithm into two sub-algorithms; i.e., IDEnc and MsgEnc. This relaxes security requirements about RevEnc components. We explain below the choices that were made.

- Encryption schemes $\mathcal{S}_{\mathrm{ID}}$ and $\mathcal{S}_{\mathrm{Msg}}$ are required to be respectively IE and IK so as to prevent that the recipient's identifier (namely, upk) leaks from ciphertext $(\mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk}), \mathsf{MsgEnc}_{\mathsf{upk}}(\mathcal{E}_k(m)))$.

Remark that IDEnc algorithm is used to output ciphertexts intended only to be decrypted by the tracing authority. This algorithm is therefore not necessarily required to be anonymous. Remark also that key $k = \mathrm{KDF}(c_1)$ is randomized because $c_1 = \mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk})$ is randomized.

- Key derivation function KDF is required to be second-preimage resistant. If this requirement is not met, it would be possible to substitute c_1 with a different c_1' having the same image through the key derivation function but disallowing the tracing of upk.

- Symmetric encryption scheme $\mathcal{E}$ is required to be non-malleable.

  Remember that in the *non-malleability* definition, the relation "$\mathsf{Trace}_{\mathsf{osk}}(\boldsymbol{c}) \neq \mathsf{upk}$" means that the tracing authority is not able to recover upk from $\boldsymbol{c}$.

  Suppose now that a valid ciphertext $c = (c_1, c_2)$ is "perturbed" into $\bar{c} = (\bar{c}_1, c_2)$ such that $\bar{c}_1$ and c_1 only differ in a few bits. In that case, the recipient can by exhaustive search correct $\bar{c}_1$ into c_1 and recover the corresponding plaintext. But, the tracing authority can in the same way recover the recipient's identifier (upk) from corrupted $\bar{c}_1$. We can therefore assume that c_1 is corrupted into $\bar{c}_1$ such that its correct value cannot be recovered by the intended recipient. The goal of the non-malleability requirement on $\mathcal{E}$ is to prevent the recipient to infer information related to m from $\mathcal{E}_k(m) = \mathsf{MsgDec}_{\mathsf{usk}}(c_2)$ because $k = \mathrm{KDF}(c_1)$ is unknown to her (and cannot be guessed).

5. APPLICATIONS

We give two applications of our generic transformation. This first one is based on a traditional public-key infrastructure (PKI) while the second one relies on the identity-based paradigm.

To simplify the presentation, we assume that the group manager (i.e., the authority in charge of setting up and maintaining the system) and the tracing authority are a single entity, called hereafter *system authority*. It is however easy to adapt the schemes to deal with two separate entities.

5.1 PKI-based solution

In [15], Tsiounis and Yung demonstrated that El Gamal encryption scheme [10] in a prime-order group $\mathbb{G}$ is IE-CPA under the decisional Diffie-Hellman assumption. Later, Bellare *et al.* [2] showed that it also achieves IK-CPA security under the same assumption. Hence, applying our generic transformation, we get a public-key encryption scheme with revocable anonymity. The description of the resulting scheme is detailed below.

Common-key generation Taking as an input some security parameter λ, a group $\mathbb{G}$ of prime order p and a generator $g \in \mathbb{G}$ are selected. The common key is $I = \{p, g\}$.

The message space is denoted by $\mathcal{M}$. Let also a non-malleable symmetric encryption scheme $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, a second-preimage resistant key-derivation function $\mathrm{KDF} : \mathbb{G} \to \mathcal{K}$, and a cryptographic hash function $H : \mathbb{G} \to \mathcal{M}$.

The secret opening key is some random element $s \in \mathbb{Z}_p$ and the corresponding public opening key is $h = g^s$.

(The public system parameters are $\{\mathcal{M}, p, g, h, \mathcal{E}, \mathrm{KDF}, H\}$.)

Key generation In order to join the system, user $\mathcal{U}_i$ randomly chooses $x_i \in \mathbb{Z}_p$ and computes $y_i = g^{x_i}$. $\mathcal{U}_i$'s public key is $\mathsf{upk}_i = \{y_i\}$ while $\mathcal{U}_i$'s secret key is $\mathsf{usk}_i = \{x_i\}$.

Encryption To send a message $m \in \mathcal{M}$ to user $\mathcal{U}_i$, one proceeds as follows:

- pick at random $r_1 \in \mathbb{Z}_p$ and compute $c_1 = (g^{r_1}, y_i \cdot h^{r_1})$;
- compute $k = \mathrm{KDF}(c_1)$, pick at random $r_2 \in \mathbb{Z}_p$, and compute $c_2 = (g^{r_2}, H({y_i}^{r_2}) \oplus \mathcal{E}_k(m))$.

The ciphertext is $c = (c_1, c_2)$.

Decryption Upon receiving ciphertext $c = (c_1, c_2)$, user $\mathcal{U}_i$ recovers plaintext m as:

- letting $c_2 = (\varphi_1, \varphi_2)$, compute $t = H({\varphi_1}^{x_i})$ using her secret key x_i;
- compute $k' = \mathrm{KDF}(c_1)$ and obtain m as $\mathcal{E}_{k'}^{-1}(t \oplus \varphi_2)$.

Tracing The intended recipient of ciphertext $c = (c_1, c_2)$ is recovered by the system authority using secret opening key s as:

- letting $c_1 = (\vartheta_1, \vartheta_2)$, compute $y' = \vartheta_2 \cdot {\vartheta_1}^{-s}$;
- check in the list of user's public keys whether there is some $y_j = y'$;
- if so, intended recipient is $\mathcal{U}_j$.

The efficiency of this scheme can be improved by choosing a single random element in $\mathbb{Z}_p$ (i.e., $r_1 = r_2$). A ciphertext is then given by $(g^{r_1}, y_i \cdot h^{r_1}, H({y_i}^{r_1}) \oplus \mathcal{E}_k(m))$.

El Gamal scheme provides data privacy and anonymity against chosen-plaintext attacks. A scheme with revocable anonymity can be similarly obtained by considering the IE-CCA Cramer-Shoup scheme [8], later proven secure in the IK-CCA sense in [2].

5.2 Identity-based solution

The previous scheme can be adapted to fit the identity-based setting. We replace El Gamal scheme with Boneh-Franklin's $\mathsf{BasicIdent}$ scheme [5]. This latter scheme further requires a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that the bilinear Diffie-Hellman assumption holds [6].

Applying our generic transformation, we get:

Common-key generation Taking as an input some security parameter λ, groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p, a generator $g \in \mathbb{G}$, and a non-degenerate bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ are selected.

The message space is $\mathcal{M}$. Let also an encoding function $\mu : \{0,1\}^* \to \mathbb{G}$, a non-malleable symmetric encryption scheme $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, a second-preimage resistant key-derivation function $\mathrm{KDF} : \mathbb{G} \to \mathcal{K}$, and a cryptographic hash function $H : \mathbb{G}_T \to \mathcal{M}$.

The secret opening keys is some random element $s \in \mathbb{Z}_p$ and the corresponding public opening key is $h = g^s$.

(The public system parameters are $\{\mathcal{M}, p, g, \hat{e}, h, \mu, \mathcal{E}, \mathrm{KDF}, H\}$.)

Key generation In order to join the system, user $\mathcal{U}_i$ obtains from the system authority her corresponding secret key $\mathsf{usk}_i = \{g_i\}$, where $g_i = \mu(\mathcal{U}_i)^s$.

Encryption To send a message $m \in \mathcal{M}$ to user $\mathcal{U}_i$, one proceeds as follows:

- pick at random $r_1 \in \mathbb{Z}_p$ and compute $c_1 = (g^{r_1}, \mu(\mathcal{U}_i) \cdot h^{r_1})$;
- compute $k = \mathrm{KDF}(c_1)$, pick at random $r_2 \in \mathbb{Z}_p$, and compute $c_2 = (g^{r_2}, H({z_i}^{r_2}) \oplus \mathcal{E}_k(m))$ where $z_i = \hat{e}(\mu(\mathcal{U}_i), h)$.

The ciphertext is $c = (c_1, c_2)$.

Decryption Upon receiving ciphertext $c = (c_1, c_2)$, user $\mathcal{U}_i$ recovers plaintext m as:

- letting $c_2 = (\varphi_1, \varphi_2)$, compute $t = H(\hat{e}(g_i, \varphi_1))$ using her secret key g_i;
- compute $k' = \mathrm{KDF}(c_1)$ and obtain m as $\mathcal{E}_{k'}^{-1}(t \oplus \varphi_2)$.

Tracing The intended recipient of ciphertext $c = (c_1, c_2)$ is recovered by the system authority using secret opening key s as:

- letting $c_1 = (\vartheta_1, \vartheta_2)$, compute $g' = \vartheta_2 \cdot {\vartheta_1}^{-s}$;
- check in the list of users whether there is some $\mathcal{U}_j$ such that $\mu(\mathcal{U}_j) = g'$;
- if so, intended recipient is $\mathcal{U}_j$.

Here too, the efficiency of the scheme can be improved by choosing a single random element in $\mathbb{Z}_p$.

6. REFERENCES

[1] A. Barth, D. Boneh, and B. Waters. Privacy in encrypted content distribution using private broadcast encryption. In G. Di Crescenzo and A. Rubin, editors, *Financial Cryptography and Data Security*, volume 4107 of *Lecture Notes in Computer Science*, pages 52–64. Springer, 2006.

[2] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[4] M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536. Springer, 1999.

[5] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[6] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. Extended abstract in Proc. of CRYPTO 2001.

[7] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

[8] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[9] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. Computing*, 30(2):391–437, 2000.

[10] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[11] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[12] M. Izabachène and D. Pointcheval. New anonymity notions for identity-based encryption. In R. Ostrovsky, R. De Prisco, and I. Visconti, editors, *Security and Cryptography for Networks*, volume 5229 of *Lecture Notes in Computer Science*, pages 375–391. Springer, 2008.

[13] A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In K. Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 181–199. Springer, 2007.

[14] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[15] Y. Tsiounis and M. Yung. On the security of ElGamal based encryption. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134. Springer, 1998.