Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis

[Published in *IEEE Transactions on Computers* **49**(9):967-970, 2000.]

Sung-Ming Yen[1] and Marc Joye[2]

[1] Laboratory of Cryptography and Information Security (LCIS)
National Central University, Chung-Li, Taiwan 320, R.O.C.
yensm@csie.ncu.edu.tw
[2] Gemplus Card International
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
marc.joye@gemplus.com

Abstract. In order to avoid fault-based attacks on cryptographic security modules (e.g., smart-cards), some authors suggest that the computation results should be checked for faults before being transmitted. In this paper, we describe a potential fault-based attack where key bits leak only through the information whether the device produces after a temporary fault a correct answer or not. This information is available to the adversary even if a check is performed before output.

Keywords. Cryptography, exponentiation, fault-based cryptanalysis, tamper resistance, interleaved modular multiplication.

1 Introduction

In order to provide better support for data protection under strong cryptographic schemes (e.g., the RSA [1] or the ElGamal [2] systems), more and more implementations based on tamper-proof devices (e.g., the smart-card) are proposed. The main reason for this trend is that smart-cards provide high reliability and security with large memory capacity and some other characteristics that conventional plastic cards do not have. The CPU in the smart-card controls the data input and output and prevents unauthorized access to the card. With special characteristics of computational ability, large memory capacity and security, a large variety of cryptographic applications benefit from the smart-card. Due to this popular usage of tamper-resistance, much attention has recently been paid regarding the security issues of cryptosystems implemented on tamper-proof devices [3–11].

This line of research re-emerged in September 1996 when a Bellcore press release [12] reported a new kind of attack, the so-called *fault-based cryptanalysis*. In the fault-based cryptanalysis model, it is assumed that when an adversary has physical access to a tamper-proof device she may purposely induce a certain type of fault into the device. Based on a set of incorrect responses from the device, due to the presence of faults, the adversary can then extract the secrets embedded in the tamper-proof device.

2 Disclaimer

The fault-based cryptanalysis model is somewhat idealized (most cards have built-in defenses against fault-based attacks [10]) and is thus controversial. Some researchers [4, 13–16] suggest that faults can be induced by ROM overwriting, EEPROM modification, gate destruction, RAM remanence, etc… To make our attack practical, we need to be able to induce some (random) faults in a given register, i.e., change its original binary status. Our requirements are strong[1] and may appear hypothetical—*We have not performed an actual successful attack on an existing product.*

However, it is our belief that any serious card manufacturer *should* envisage that an adversary might be able to bypass the built-in defenses and analyze the effects of this intrusion. In that context, we will show—and this the main point of our paper—that checking the computation results for faults does not necessarily prevent an adversary from learning the secrets embedded in a smart-card.

3 New Fault-Based Attacks

Given the nature of hardware fault-based cryptanalysis, an engineering approach for attacking, we need to take a closer look at some particular implementation details to understand and develop new types of attacks.

3.1 Cryptosystem Implementations

Below is the commonly used algorithm for computing $M^d \bmod N$ where the exponent d is expressed in binary form as $d = \sum_{i=0}^{n-1} d_i\, 2^i$. This method is usually referred to as the right-to-left binary exponentiation algorithm [17].

In this algorithm, each iteration requires one modular squaring plus one modular multiplication depending on the bit value of d_i. In most implementations, both the multiplication and squaring operations are treated by the same routine in order to simplify and reduce the code. Furthermore, for efficiency reasons, the multiplication and reduction operations are usually interleaved [18–20].

Suppose that the modular multiplication $R = AB \bmod N$ has to be performed. Let $A = \sum_{i=0}^{n-1} a_i\, 2^i$ be the binary expansion of A. Then, grouping t bits into a single memory word, A can be recoded in radix 2^t as $A = \sum_{j=0}^{m-1} A_j\,(2^t)^j$ with $m = \lceil n/t \rceil$. Hence we can write the product of A and B as

$$\begin{aligned} R = ((\cdots((A_{m-1}B)2^t + A_{m-2}B)2^t + \cdots + A_1B)2^t \\ +A_0B) \bmod N, \end{aligned}$$

[1] Certain attacks appearing in the literature assumes much stronger requirements.

```
Input:  $M, d = (d_{n-1}, \dots, d_0)_2, N$
Output: $A = M^d \bmod N$
─────────────────────────────
1.1 $A \leftarrow 1$; $B \leftarrow M$
1.2 for $i$ from $0$ to $n-1$
1.3     if $(d_i = 1)$ then $A \leftarrow A \cdot B \bmod N$
1.4     $B \leftarrow B^2 \bmod N$
1.5 endfor
```

Fig. 1. Right-to-left binary exponentiation – Algorithm 1.

or in algorithmic form:

```
Input:  $A, B, N$
Output: $R = AB \bmod N$
─────────────────────────────
2.1 $R \leftarrow 0$
2.2 for $j$ from $m-1$ downto $0$
2.3     $R \leftarrow (R\,2^t + A_j B) \bmod N$
2.4 endfor
```

Fig. 2. Radix 2^t interleaved modular multiplication – Algorithm 2.

Replacing the modular multiplications (Lines 1.3 and 1.4 in Algorithm 1) by the previous procedure, Algorithm 1 can thus be rewritten in a more detailed way as follows.

3.2 Proposed Attack

To simplify the discussion, we will assume that the aim of an adversary is to find the secret value of exponent d involved in the evaluation of $M^d \bmod N$. (Think for example that d is a secret RSA decryption or signature key.) However, we note that similar ideas may be applied to recover secret parameters in more complex computations.

The basic idea of the proposed attack relies on the observation that after the computation of $AB \bmod N$, the result is re-assigned to register A (Line 3.8 in Algorithm 3). Therefore, if one or several bits of error are introduced into the more significant bit positions of register A, *no* error will be detected after restoring the result R into A if the faulty bits belong to words A_j are no longer required. More precisely, suppose for example that during iteration i of the main `for-loop` (Lines 3.2–3.15), some bits of word A_k are maliciously modified. If bit d_i is equal to 1 and if the error is induced when counter j of the subsequent inner `for-loop` (Lines 3.5–3.7) is less than k, then the modified A_k will not damage

```
Input:  $M, d = (d_{n-1}, \ldots, d_0)_2, N$
Output: $A = M^d \bmod N$

3.1  $A \leftarrow 1$; $B \leftarrow M$
3.2  for $i$ from $0$ to $n-1$
3.3    if $(d_i = 1)$ then
3.4      $R \leftarrow 0$
3.5      for $j$ from $m-1$ downto $0$
3.6        $R \leftarrow (R\,2^t + A_j B) \bmod N$
3.7      endfor
3.8      $A \leftarrow R$
3.9    endif
3.10   $R \leftarrow 0$
3.11   for $j$ from $m-1$ downto $0$
3.12     $R \leftarrow (R\,2^t + B_j B) \bmod N$
3.13   endfor
3.14   $B \leftarrow R$
3.15 endfor
```

Fig. 3. (Interleaved) Right-to-left exponentiation – Algorithm 3.

the correctness of the final value of R. Eventually, after the restoring operation $A \leftarrow R$ (Line 3.8), the error located in register A will be cleared. Such kind of temporary error will be called *safe error*. However, if an error is introduced in word A_k while bit d_i is equal to 0 then register A is not cleared and the final value will be incorrect. We thus have a simple means to know the value of bit d_i.

After a very short period of initialization for some hardware registers and RAM memory, the right-to-left exponentiation algorithm (Algorithm 1) performs a sequence of modular multiplications

$$\{O_1, O_2, O_3, O_4, \ldots, O_q\}$$

where q is the sum of n and the Hamming weight of d.

The following cryptanalysis deriving the secret information d begins from the extraction of d_0 through d_{n-1}. Of course, random order extraction of bits d_i is possible if required. During each bit derivation, the attacker guesses that $d_i = 1$. If d_i is effectively equal to 1, then the exponentiation algorithm will compute both

$$A \leftarrow A \cdot B \bmod N \qquad \text{(denoted as } O_{mult}\text{)}$$

and

$$B \leftarrow B^2 \bmod N \qquad \text{(denoted as } O_{squ}\text{)}$$

in that order. Then the attacker introduces the previously mentioned *safe error* into register A during operation O_{mult} being executed and brings no further interruption to the hardware. The correctness of the guess can be verified via the output generated by the hardware. Taking again the RSA cryptosystem for demonstration purpose, the attacker can raise the final output A to the eth power as $A^e \bmod N$, where e is the public key corresponding to secret key d. If this value is equal to the original number M, then the secret bit d_i being guessed is indeed equal to 1, otherwise $d_i = 0$. This follows from the fact that if $d_i = 0$, the operation O_{mult} is bypassed and the introduced error into register A will no longer be *safe*.

Almost all research results regarding fault-based cryptanalysis conclude that the computations should be checked in order to prevent possible fault-based attacks. The most interesting thing in the above attack is that even the hardware is designed to refuse to release incorrect results, the attacker still gains the exact knowledge of d_i because he knows that, in that case, the introduced error is not *safe* and thus that $d_i = 0$. This clearly shows that checking before output does not necessarily thwart fault-based attacks.

Remark: Someone may argue that the attack can easily be defeated if the hardware re-calculates its output when it detects a fault (note that this means that the hardware releases an output in every case). However, such faults can still be detected by use of a timing attack. When the hardware re-calculates a value after it detects an 'un-safe' fault it will take twice as long to output an answer, and this should be glaringly obvious. Therefore, when the hardware takes twice as long as usual to produce an output, we can deduce that an *un-safe error* must have occured and proceed as before to conclude that d_i must be 0.

The following procedure summarizes the attack to recover the secret exponent d.

```
$\ell \leftarrow 1$
for $i$ from 0 to $n-1$
  Introduce a safe error into $A$ during $O_\ell$
        and complete the exponentiation
  if (output refused or incorrect) then
    $d_i \leftarrow 0$; $\ell \leftarrow \ell + 1$
  else
    $d_i \leftarrow 1$; $\ell \leftarrow \ell + 2$
  endif
endfor
```

Fig. 4. Proposed attack (right-to-left version).

3.3 Feasibility of the Attack

Three classes of hardware fault-based cryptanalysis can basically be distinguished: the first one assumes a very precise controllability of *fault location*, the second one only needs a loose controllability while the third one assumes absolutely no control over the location of the fault. In fact, some minimum required controllability of fault location is always needed in order to induce a fault in an exact register. Often, an attack with more precise controllability of fault location can be achieved with less computation and fewer interactions with the hardware. In the previous attack, the fault location assumption is not very restricted. It can be traded off with the following assumption of *fault occurrence time*, e.g., the moment when the multiplication (Line 1.3 in Algorithm 1) is performed (an assumption made in some existing attacks) or even, more precisely, the moment when the interleaved multiplications (Line 3.6 in Algortihm 3) are performed. Of course, such extremely precise timing controllability may be conceivable when the attacker has the overall control of the hardware. For the timing controllability, it is important to notice that the *clock* signal of current smart IC cards is supplied from the card reader. Our attack, however, does not require a very precise timing controllability. Only an approximation on the time τ required to perform a modular multiplication O_ℓ is needed. With this timing estimation and the parameter m, a loose timing controllability of each interleaved multiplication (Line 3.6) is possible.

As mentioned before, the total number q of modular multiplications to be performed is the sum of n and the Hamming weight of d, q is thus equal to $1.5n$, on average. Therefore a good estimation on τ can be easily obtained after a few experiments (on some different cards) by dividing the time to compute $M^d \bmod N$ by $1.5n$. After obtaining τ, the tradeoff between *fault location* and *fault occurrence time* goes as follows. If a more precise τ (and thus a more precise timing controllability) is available, then it is more feasible to predict the value of counter j of the inner `for-loop` (Lines 3.5–3.7) at any moment. This follows from the fact that in each time period τ there are m modular operations $R \leftarrow (R\,2^t + A_j B) \bmod N$ to be performed; each operation taking τ/m second. This more precise prediction of course relaxes, to some degree, the requirement of precise controllability of *fault location*. For example, when the adversary knows $j = k$, then he can introduce an error among words A_ℓ $(m-1 \le \ell < k)$. On the contrary, if the adversary possesses some techniques to introduce error at a precise location (now he prefers the more significant positions of A), he can therefore conduct a more loose control of timing.

About the classification of faults assumed in the fault-based cryptanalysis, the fault type and the bit length of fault can be two good viewpoints. For the problem of fault type, previously existing attacks assume a temporary fault to be one of: stuck at 1 or 0 fault, flipping fault, or just random fault. Clearly, the random fault model is the most general assumption and will make an attack more practical. In the safe error based attack proposed in this paper, we assume only the existence of random faults. From the viewpoint of bit length of the error, both single-bit fault and multi-bit fault have been assumed in previously existing attacks. Generally speaking, it is much difficult to induce a single-bit fault precisely than to induce a block of faulty bits. The proposed safe error based attack does not limit how many bits of fault should be induced into the register A. The only requirement is that the bits to be corrupted belong to words A_j that are no longer required.

3.4 Speeding up the Attack

For some special cases, the recovery of exponent d can be speeded up. Once again, we will use RSA for the illustration purpose. In RSA, the secret exponent d and the public exponent e satisfy $ed \equiv 1 \pmod{\phi(N)}$, where ϕ is the Euler's totient function; or equivalently, there exists an integer k such that $ed - k\phi(N) = 1$. Since $d < \phi(N)$, we have $k < e$. Letting $\tilde{d} = \lfloor \frac{1+k(N+1)}{e} \rfloor$, a trivial argument shows that the $n/2$ topmost bits of $\tilde{d}$ and d are the same [22, Proof of Fact 3.2]. So, for low exponent e, the attacker can try each candidate $k < e$, compute the corresponding $\tilde{d}$ and recover the $n/2$ topmost bits of d if the correct value for k is guessed. This guess can be checked from the knowledge of the $n/2$ least significant bits of d.

Using a powerful technique due to Coppersmith [21], Boneh *et al.* [22] improved this bound and pointed out that only the $n/4$ *least significant bits* of d suffice to recover the entire exponent d in the case of a low exponent e. On the other hand, for "large" values of e, they showed that, given the factorization of e and (at most) $n/2$ *most significant bits* of d, the entire secret exponent d can also be recovered.

4 Extension to Other Implementations

Although we demonstrate the attack under the right-to-left exponentiation technique in the previous section, it can be easily verified that the attack still works when the left-to-right exponentiation technique is employed for computing $M^d \bmod N$ [17].

```
Input:  $M, d = (d_{n-1}, \dots, d_0)_2, N$
Output: $A = M^d \bmod N$
-----------------------------------------
5.1 $A \leftarrow 1$
5.2 for $i$ from $n-1$ downto 0
5.3     $A \leftarrow A^2 \bmod N$
5.4     if $(d_i = 1)$ then $A \leftarrow A \cdot M \bmod N$
5.5 endfor
```

Fig. 5. Left-to-right binary exponentiation – Algorithm 5.

It is important to note that when an error is introduced to register A during the operation $A \leftarrow A^2 \bmod N$, it will force the squaring operation to be incorrect. This is evident because the correct value of A is required during each iteration of the interleaved modular multiplication procedure. However, if a *safe error* is introduced into A during the operation $A \leftarrow A \cdot M \bmod N$, then this error will not damage the final result. The above attack is sketched hereafter.

```
$\ell \leftarrow 2$
for $i$ from $n-1$ downto $0$
  Introduce a safe error into $A$ during $O_\ell$
                and complete the exponentiation
  if (output refused or incorrect) then
    $d_i \leftarrow 0$; $\ell \leftarrow \ell + 1$
  else
    $d_i \leftarrow 1$; $\ell \leftarrow \ell + 2$
  endif
endfor
```

Fig. 6. Proposed attack (left-to-right version).

Furthermore, when other types of interleaved multiplication algorithms scanning the multiplier from the least significant position are used, the attack can still be modified to work easily.

5 Extension to Symmetric Cryptosystems

In [8], Biham and Shamir extended the Bellcore attack to an extremely different branch: they considered fault-based cryptanalysis on symmetric cryptosystems, e.g., the DES [23]. It is called the *differential fault analysis* (DFA) and it seems to be applicable to almost all symmetric cryptosystems.

It might be worthwhile to notice that the potential attack described in this paper can be extended to symmetric cryptosystems. The concept of safe error, under the assumption that an adversary has only the knowledge of error or error free from the hardware device, can be applied to these systems as well. The theoretical work on the extension and exact cryptanalytic process for specific systems are still under construction. These future research results, if proven to be of practical value, will bring new understanding of precautions for symmetric cryptosystems implemented within tamper-proof devices.

6 Concluding Remarks

In this paper, we demonstrate one type of new and powerful hardware fault-based attack based on the proposed *safe error* concept. These attacks (assuming the fault-based cryptanalysis model, see Section 2) are shown to be powerful because the cryptanalytic complexities, especially the computational complexity, are quite small compared with other existing attacks. The purpose is to show that checking the correctness of the computed result before giving it to others may not be enough to prevent a hardware fault-based cryptanalysis. We not only propose new attacks but also provide motivations for researchers and developers in this field working on this rapidly growing important topic. However, this does not imply that it is not possible or difficult to withstand such kind of new attacks, at least for the attacks considered in this paper. One simple solution, using the right-to-left binary exponentiation for example, is to let register B play the role of register A (to be as the *multiplier*) in the interleaved modular multiplication procedure.

Since the hardware fault-based cryptanalysis is in essence an engineering oriented cryptanalysis, the authors suggest cryptographic hardware designers to carefully consider each possible implementation detail when developing a secure system.

Acknowledgments

We would like to thank the anonymous referees for their useful comments. In particular, a referee pointing out the possibility to extend the attack to symmetric cryptosystems is highly appreciated.

References

1. R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystem," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
2. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, July 1985.
3. D. Boneh, R.A. DeMillo, and R.J. Lipton, "On the importance of checking cryptographic protocols for faults." In *Advances in Cryptology – EUROCRYPT '97*, LNCS 1233, pp. 37–51, Springer-Verlag, Berlin, 1997.
4. R. Anderson and M. Kuhn, "Tamper resistance – a cautionary note." In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pp. 1–11, USENIX Association, Berkeley, 1996.
5. M. Joye, A.K. Lenstra, and J.-J. Quisquater, "Chinese remaindering based cryptosystems in the presence of faults." To appear in Journal of Cryptology.
6. F. Bao, R.H. Deng, Y. Han, A. Jeng, A.D. Narasimbalu, and T. Ngair, "Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults." In *Security Protocols*, LNCS 1361, pp. 115–124, Springer-Verlag, Berlin, 1998.
7. 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*, Fukuoka, 29th Jan.–1st Feb. 1997.
8. E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems." In *Advances in Cryptology – CRYPTO '97*, LNCS 1294, pp. 513–525. Springer-Verlag, Berlin, 1997.
9. A. Shamir, "How to check modular exponentiation." Presented at the rump session of EUROCRYPT '97, Konstanz, 11–15th May 1997.
10. D.P. Maher, "Fault induction attacks, tamper resistance, and hostile reverse engineering in perspective." In *Financial Cryptography*, LNCS 1318, pp. 109–121, Springer-Verlag, Berlin, 1997.

11. B.S. Kaliski Jr. and M.J.B. Robshaw, "Comments on some new attacks on cryptographic devices," RSA Laboratories' Bulletin, no. 5, RSA Laboratories, Redwood City, July 1997.
12. Bellcore Press Release, "New threat model breaks crypto codes," Sept. 1996.
13. R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices." In *Security Protocols*, LNCS 1361, pp. 125–136, Springer-Verlag, Berlin, 1998.
14. 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.
15. O. Kocar, "Hardwaresicherheit von Mikrochips in Chipkarten," *Datenschutz und Datensicherheit*, vol. 20, no. 7, pp. 421–424, July 1996.
16. I. Peterson, "Chinks in digital armor—Exploiting faults to break smart-card cryptosystems," *Science News*, vol. 151, no. 5, pp. 78–79, Feb. 1997.
17. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, Chapter 14, CRC Press, New York, 1997.
18. G.R. Blakley, "A computer algorithm for the product AB modulo M," *IEEE Transactions on Computers*, vol. 32, no. 5, pp. 497–500, May 1983.
19. K.R. Sloan, Jr., Comments on "A computer algorithm for the product AB modulo M," *IEEE Transactions on Computers*, vol. 34, no. 3, pp. 290–292, March 1985.
20. Ç.K. Koç, "RSA hardware implementation," Technical Report TR 801, RSA Laboratories, Redwood City, April 1996
21. D. Coppersmith, "Finding a small root of a univariate modular equation." In *Advances in Cryptology – EUROCRYPT '96*, LNCS 1070, pp. 155–165, Springer-Verlag, Berlin, 1996.
22. D. Boneh, G. Durfee, Y. Frankel, "An attack on RSA given a small fraction of the private key bits." In *Advances in Cryptology – ASIACRYPT '98*, LNCS 1514, pp. 25–34, Springer-Verlag, Berlin, 1998.
23. NBS FIPS PUB 46, "Data Encryption Standard," National Bureau of Standard, U.S. Department of Commerce, Jan. 1977.