A Survey on Fault-Based Attack to RSA

Xun Li Cetin Kaya Koc

Abstract— **RSA has been used as one of the most popular symmetric encryption algorithm since decades ago. The security policies of a substantial amount of devices, protocols and systems today depend on the RSA algorithm. Hence the reliability of the RSA algorithm is crucial to today's computer infrastructures. Although there is no explicit weakness in RSA, different ingenious side-channel attacks have been found out to extract the secrete keys. Among those attacks, fault-based attacks have been previously known as mainly theoretical and impractical to implement. However, recent studies show that it is feasible to apply fault-based attacks to extract the secrete keys of RSA through fault injection. In this paper, we give a survey on the field of fault-based attack targeting to the RSA algorithm, and discuss different approaches.**

I. Introduction

The RSA cryptosystem was initially invented by Ron Rivest, Adi Shamir, and Len Adleman [1] in 1977. As one of the most popular public key algorithms, RSA has been widely used in many commercial systems, such as web servers, login systems and bank verification systems. Any flaw in the RSA algorithm could potentially case the failure of many different authentication systems in the world. Due to its popularity, the RSA algorithm has been analyzed for decades. Since the first work on timing attack to RSA algorithm published by Kocher [2] in 1996, different side-channel attacks have been invented to break the RSA cryptosystem without factoring the modulus. Among those, fault-based attacks have been shown as the most powerful and practical approaches.

The RSA algorithm works as follows: Let $N = pq$ be a product of two large primes each $\frac{n}{2}$ bits long. To sign a message M, one computes

$$S = M^d \ mod \ N \tag{1}$$

where d is the secrete key. To verify the message, one computes

$$M = S^e \ mod \ N \tag{2}$$

where e is the public key. For efficiency, the Chinese Remainder Theory (CRT) is commonly used in the RSA algorithm: One first compute

$$S_1 = M^d \ mod \ p \tag{3}$$

and

$$S_2 = M^d \ mod \ q \tag{4}$$

then use the constants obtained by CRT to compute

$$S = M^d \ mod \ N \tag{5}$$

Authors are with the Department of Computer Science, University of California, Santa Barbara, CA 93106. E-mail: {xun,koc}@cs.ucsb.edu

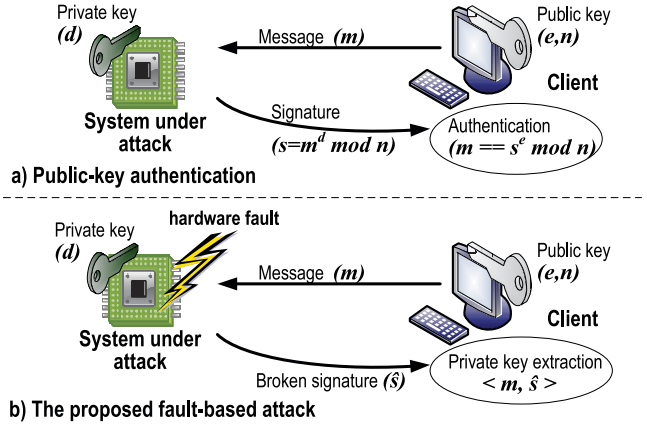

Fig. 1. RSA mechanism and proposed attack

The number of bits operated drops by half. Figure 1(a) shows how the RSA encryption and decryption works.

In this paper, we will survey previous fault-based attacks on RSA algorithm and their countermeasures.

II. Attacks and Defences

A. First Attack by Boneh

Initially published in 1997 by Boneh et al [3], a fault-based attack can be easily performed on the CRT based RSA algorithm. The idea here is that given a faulty message generated by the RSA algorithm during which some random error happened, it is sufficient to factor the N. Let $x \in \mathbb{Z}_{\mathbb{N}}$ be a message and let $S = x^d \ mod \ N$ be a valid RSA signature of x. Let $\hat{S}$ be a faulty signature. Recall that S is computed by first computing S_1 and S_2. Similarly, $\hat{S}$ is computed by first computing $\hat{S}_1$ and $\hat{S}_2$. Suppose that during the computation of $\hat{S}$ an error occurs during the computation of only one of $\hat{S}_1$, $\hat{S}_2$. Without loss of generality, suppose a hardware fault occurs during the computation of $\hat{S}_1$ (i.e. $S_1 \neq \hat{S}_1 \ mod \ p$) but no fault occurs during the computation of $\hat{S}_2$ (i.e. $\hat{S}_2 = S_2$). Then $S = \hat{S} \ mod \ q$, but $S \neq \hat{S} \ mod \ p$. Therefore,

$$gcd(S - \hat{S}, N) = q \tag{6}$$

and so N can be easily factored. Figure 1(b) shows how such attack can be performed.

B. Shamir's Countermeasure

The main idea to defence the above attack is to check whether the result is correct before output, such that error messages will never be seen by the attacker. Shamirs idea [4] is to select a random integer t and to do the following computations:

$$S_{pt} := m^d \ mod \ p * t \tag{7}$$

$$S_{qt} := m^d \ mod \ q * t \tag{8}$$

In the case of $S_{pt} = S_{qt} \ mod \ t$ the computation is defined to be error free and S is computed according to the CRT recombination equation.

One drawback in Shamirs method, as pointed out in [5], is the following: Within the CRT mode of real RSA applications the value d is not known, only the values $d_p = d \ mod \ (p-1)$ and $d_q = d \ mod \ (q-1)$ are known. Although d can be efficiently computed from d_p and d_q only, it will limit the acceptance of Shamirs method.

C. Attacks against Shamir's

As pointed out by [6], the problem with Shamir's countermeasure is then, the checking strategy can only guarantee that no errors have happened at the point of S_{pt} and S_{qt} are calculated. There can still be errors generated in the process of calculating the final signature. After the computation of S_{pt} and S_{qt}, partial signatures are computed:

$$S_p = S_{pt} \ mod \ p \tag{9}$$

$$S_q = S_{qt} \ mod \ q \tag{10}$$

If by accident, a faulty result of S_p is obtained when S_{pt} is reduced modulo p and all other intermediate results are correct. Then, the RSA modulus n can be factorized by the following two approaches. In the first case, if the attacker can obtain both the correct signature s and the erroneous signature $\hat{s}$(by recombining $\hat{s_p}$ and s_q), then the following computation can factorize n:

$$q = gcd(\hat{s} - s, n) \tag{11}$$

where the erroneous signature

$$\hat{s} = s_q + ((\hat{s_p} - s_q) \cdot (q - 1 \ mod \ p) \ mod \ p) \cdot q \tag{12}$$

In the second case, if the attacker can only obtain the erroneous signature $\hat{s}$ and its related message m, then the following computation can factorize n:

$$q = gcd(\hat{s}^e - m, n) \tag{13}$$

D. Aumüller's Countermeasure

To counter the above attack, Aumüller [7] proposed additional checking algorithms during the stage of combining partial signatures to generate the final signature. The countermeasure algorithm is shown in Figure 2. They also showed how practical fault attacks countermeasures can be implemented.

E. Yen's Countermeasure

In [8], Yen et al. note that error detection based on decisional tests should be avoided. Indeed, inducing a random fault in the status register flips the value of the zero flag bit with a probability of 50% and so bypasses the error detection in the case of a faulty computation. Starting from this observation, they introduce the concept of infective computation. The attack described in previous sections requires that only a half exponentiation is faulty but not both. The idea behind infective computation is to ensure that both half exponentiations are faulty whenever an error is induced: if $\hat{S} \equiv S \ (mod \ p)$ then $\hat{S} \equiv S \ (mod \ q)$, and conversely.

input: $m, p, q, d_p, d_q, q^{-1} \bmod p$

let t be a short prime number, e.g., 32 bits

$p' := p * t$
$d'_p := d_p + \text{random}_1 * (p-1)$
$S'_p := m^{d'_p} \bmod p'$
if $\neg(p' \bmod p \equiv 0 \ \wedge \ d'_p \bmod (p-1) \equiv d_p)$ **then return**(error)

$q' := q * t$
$d'_q := d_q + \text{random}_2 * (q-1)$
$S'_q := m^{d'_q} \bmod q'$
if $\neg(q' \bmod q \equiv 0 \ \wedge \ d'_q \bmod (q-1) \equiv d_q)$ **then return**(error)

$S_p := S'_p \bmod p$
$S_q := S'_q \bmod q$
$S := S_q + ((S_p - S_q) * q^{-1} \bmod p) * q$
if $\neg((S - S'_p \bmod p \equiv 0) \ \wedge \ (S - S'_q \bmod q \equiv 0))$ **then return**(error)

$S_{pt} := S'_p \bmod t$
$d_{pt} := d'_p \bmod (t-1)$
$S_{qt} := S'_q \bmod t$
$d_{qt} := d'_q \bmod (t-1)$
if $(S_{pt}^{d_{qt}} \equiv S_{qt}^{d_{pt}} \bmod t)$ **then**
  return(S)
else
  return(error)

output: $m^d \bmod (p * q)$

Fig. 2. Aumüller's Countermeasure

F. Ciet's Countermeasure

Unfortunately, the above countermeasure is shown to be insecure. The attack exploits a transient single-byte fault that modifies the value of m as it is read in memory in the computation of s_p^* but leaves its value stored in memory unaccepted.

Their countermeasures start with the generalized Shamir's trick and adapt it to avoid decisional tests using the infective computation methodology of [8]. More precisely, for two security parameters κ and ι, the device computes

$$i_{q^*} = (r_2 q)^{-1} \ mod \ (r_1 p) \tag{14}$$

$$s_p^* = m^{d_p} \ mod \ (r_1 p) \tag{15}$$

$$s_1 = m^{d_p \ mod \ \varphi(r_1)} \ mod \ r_1 \tag{16}$$

$$s_q^* = m^{d_q} \ mod \ (r_2 q) \tag{17}$$

$$s_2 = m^{d_q \ mod \ \varphi(r_2)} \ mod \ r_2 \tag{18}$$

$$\tag{19}$$

where r_1 and r_2 are two co-prime random κ-bit integers, and returns the signature

$$S = (S^*)^\gamma \ mod \ N \tag{20}$$

where the detailed definition can be found in the paper.

G. Practical Attacks

Although there have been so many different attacks proposed in the past decade, none of them have been practically applied to modern computers and retrieve secrete information. The first practical implementation of fault based attack on RSA was proposed in 2010 by Pellegrini et. al. [9]. In this work the authors detail a complete end-to-end fault-attack on a microprocessor system and practically demonstrate how hardware vulnerabilities can be exploited to target secure systems. They developed a theoretical attack to the RSA signature algorithm, and realized it in practice against an FPGA implementation of the system under attack. To perpetrate the attack, they inject transient faults in the target machine by regulating the voltage supply of the system. Thus, such attack does not require access to the victim systems internal components, but simply proximity to it.

III. Conclusions

We have seen many different ingenious ways to perform fault based attack on the RSA algorithm. The RSA algorithm, supported by other optimization methods like CRT is complicated enough to expose infinite number of weakness for fault based attack. And the introduction of every kind of countermeasure always generate new possibilities for different fault based attacks. Starting from the year of 2010 during which the first practical fault based attack targeting to microprocessor systems was achieved, understanding the attacks and countermeasures on RSA algorithm becomes more important. We look forward in the future that a fault-based-attack-free RSA algorithm can be found.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. of the ACM*, pp. 120–126, 1978.

[2] Paul C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, London, UK, 1996, pp. 104–113, Springer-Verlag.

[3] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton, "On the importance of eliminating errors in cryptographic computations," *JOURNAL OF CRYPTOLOGY*, pp. 101–119, 2001.

[4] A. Shamir, "Method and apparatus for protecting public key schemes from timing and fault attacks," pp. 375, 377, 381, November 1999.

[5] Marc Joye, Pascal Paillier, and Sung-Ming Yen, "Secure evaluation of modular functions," 2001.

[6] Sung-Ming Yen, Sang-Jae Moon, and JaeCheol Ha, "Hardware fault attackon rsa with crt revisited," in *ICISC*, 2002, pp. 374–388.

[7] C. Aumüler, P. Bier, W. Fischer, P. Hofreiter, J. p. Seifert, and Chipcard Ics, "Fault attacks on rsa with crt: Concrete results and practical countermeasures," 2002.

[8] S.-M. Yen, S. Kim, S. Lim, and S. Moon, "Rsa speedup with residue number system immune against hardware fault cryptanalysis," *K. Kim, editor, Information Security and Cryptology - ICISC 2001*, vol. 2288 of Lecture Notes in Computer Science, pp. 397.

[9] A. Pellegrini, V. Bertacco, and T. Austin, "Fault-based attack of rsa authentication," *Proc. Design and Test*, 2010.