Fault Attacks Against EMV Signatures

Jean-Sébastien Coron[1], David Naccache[2], and Mehdi Tibouchi[2]

[1] Université du Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg, Luxembourg
{jean-sebastien.coron, avradip.mandal}@uni.lu
[2] École normale supérieure
Département d'informatique, Groupe de Cryptographie
45, rue d'Ulm, F-75230 Paris CEDEX 05, France
{david.naccache, mehdi.tibouchi}@ens.fr

Abstract. At CHES 2009, Coron, Joux, Kizhvatov, Naccache and Paillier (CJKNP) exhibited a fault attack against RSA signatures with partially known messages. This attack allows factoring the public modulus N. While the size of the unknown message part (UMP) increases with the number of faulty signatures available, the complexity of CJKNP's attack increases exponentially with the number of faulty signatures.
This paper describes a simpler attack, whose complexity is polynomial in the number of faults; consequently, the new attack can handle much larger UMPs. The new technique can factor N in a fraction of a second using ten faulty EMV signatures – a target beyond CJKNP's reach.
We show how to apply the attack even when N is unknown, a frequent situation in real-life attacks.

Keywords: Fault Attacks, Digital Signatures, RSA, ISO/IEC 9796-2, EMV.

1 Introduction

RSA [21] is certainly the most widely used signature scheme. To sign a message m with RSA, the signer first applies an encoding function μ to m, and then computes the signature $\sigma = \mu(m)^d \bmod N$. To verify the signature, the receiver checks that

$$\sigma^e = \mu(m) \mod N.$$

The Chinese Remainder Theorem (CRT) is often used to reduce the signer's computational load. This is done by computing:

$$\sigma_p = \mu(m)^d \mod p \quad \text{and} \quad \sigma_q = \mu(m)^d \mod q$$

and the signature σ is computed from σ_p and σ_q by Chinese Remaindering.

In [2], Boneh, DeMillo and Lipton showed that RSA implementations can be vulnerable to fault attacks (see also [15]). Assuming that the attacker can induce a fault when σ_q is computed while keeping the computation of σ_p correct, one gets

$$\sigma_p = \mu(m)^d \mod p \quad \text{and} \quad \sigma_q \neq \mu(m)^d \mod q$$

and the resulting (faulty) signature σ satisfies

$$\sigma^e = \mu(m) \mod p \quad \text{and} \quad \sigma^e \neq \mu(m) \mod q\ .$$

Whereby the attacker can then factor N by

$$\gcd(\sigma^e - \mu(m) \bmod N, N) = p\ . \tag{1}$$

It is easy to see that Boneh *et al.*'s fault attack applies to any deterministic RSA encoding, *e.g.* the Full Domain Hash (FDH) [1] encoding where

$$\sigma = H(m)^d \mod N$$

and $H : \{0,1\}^* \mapsto \mathbb{Z}_N$ is a hash function. The attack is also applicable to probabilistic signature schemes where the randomizer used to generate the signature is sent along with the signature.

However, if the randomizer is only recovered when verifying the signature, or if some part of the message is unknown, the attack is thwarted. For example, consider a signature $\sigma = (m\|r)^d \bmod N$. The random r is only be recovered when verifying a *correct* signature. Given a faulty signature, the attacker cannot retrieve r nor infer $(m\|r)$ which would be necessary to compute

$$\gcd(\sigma^e - (m\|r) \bmod N, N) = p.$$

At CHES 2009, Coron, Joux, Kizhvatov, Naccache and Paillier (CJKNP) showed how to extend Boneh *et al.*'s attack to RSA signatures with partially unknown messages (or unknown nonces) [4]. CJKNP's attack was illustrated with a probabilistic variant of the ISO/IEC 9796-2 standard [13], as used in the EMV specifications [10]. In ISO/IEC 9796-2 the encoded message has the form:

$$\mu(m) = \mathtt{6A}_{\mathtt{16}} \,\|\, m[1] \,\|\, H(m) \,\|\, \mathtt{BC}_{\mathtt{16}}$$

where $m = m[1] \,\|\, m[2]$ is split into two parts. CJKNP show that if the unknown part of $m[1]$ is not too large (*e.g.* less than 160 bits for a 2048-bit RSA modulus), then a single faulty signature allows to factor N as in Boneh *et al.*'s attack. CJKNP's attack is based on a technique due to Herrmann and May [9] for finding small roots of linear equations modulo an unknown factor p of N; [9] is itself based on Coppersmith's technique [3] for finding small roots of polynomial equations using the LLL algorithm [18]. In addition, [4] introduced a multi-fault attack using an extension of Coppersmith's attack. Multiple faults make it possible to attack larger unknown message parts (UMPs). However, this comes at the cost of a complexity *exponential* in the number of faults.

This paper describes a simpler multiple fault attack. The new attack's complexity is polynomial in the number of faulty signatures. This allows us to tackle larger UMPs which were beyond [4]'s reach. For example, in a typical use case of EMV, ten faulty signatures are enough to factor N in a fraction of a second with our new attack, whereas the attack in [4] was completely impractical in such a situation.

Finally, we show that a similar technique can even recover N from a collection of valid signatures, so that the attack can be appplied to protocols in which *public* RSA parameters are not available to outsiders, which do arise in practice (*e.g.* proprietary banking cards or e-passports).

2 Coron-Joux-Kizhvatov-Naccache-Paillier's Attack

2.1 ISO/IEC 9796-2 Standard with Partially Unknown Message

[4] considers a randomized version of the ISO/IEC 9796-2 standard. ISO/IEC 9796-2 is an encoding standard allowing partial or total message recovery [13, 14]. The encoding can be used with hash

functions $H(m)$ of various digest sizes k_h. When k_h, the size of m and the size of N (denoted k) are all multiples of 8, the ISO/IEC 9796-2 encoding of a message $m = m[1] \,\|\, m[2]$ is

$$\mu(m) = \mathtt{6A}_{16} \,\|\, m[1] \,\|\, H(m) \,\|\, \mathtt{BC}_{16}$$

where $m[1]$ consists of the $k - k_h - 16$ leftmost bits of m and $m[2]$ represents the remaining bits of m. In [14] it is required that $k_h \geq 160$. This is also the case in the EMV specifications [10]. We note that a practical forgery attack (without faults) against ISO/IEC 9796-2 was recently described in [6], extending the attack in [5]. However the attack is only practical when $m[1]$ can be fully chosen by the adversary, which is not the case in EMV and in the randomized version of the ISO/IEC 9796-2 standard considered in this paper.

More precisely, [4] considers a message $m = m[1] \,\|\, m[2]$ of the form

$$m[1] = \alpha \,\|\, r \,\|\, \alpha', \quad m[2] = \text{DATA}$$

where r is a message part unknown to the adversary (UMP), α and α' are strings known to the adversary and DATA is some known or unknown string. The size of r is denoted by k_r and the size of $m[1]$ is $k - k_h - 16$ as required in ISO/IEC 9796-2. The encoded message is then

$$\mu(m) = \mathtt{6A}_{16} \,\|\, \alpha \,\|\, r \,\|\, \alpha' \,\|\, H(\alpha \,\|\, r \,\|\, \alpha' \,\|\, \text{DATA}) \,\|\, \mathtt{BC}_{16} \tag{2}$$

Therefore both r and $H(\alpha \,\|\, r \,\|\, \alpha' \,\|\, \text{DATA})$ are unknown; the total number of unknown bits inside $\mu(m)$ is then $k_r + k_h$.

2.2 Single Fault Attack

[4] describes a fault attack against the previous signature scheme. More precisely, one assumes that after injecting a fault the opponent has a faulty signature σ such that:

$$\sigma^e = \mu(m) \mod p\,, \quad \sigma^e \neq \mu(m) \mod q\,. \tag{3}$$

From (2) one can write

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8 \tag{4}$$

where t is a known value. From (3) one gets:

$$\sigma^e = t + r \cdot 2^{n_r} + H(m) \cdot 2^8 \mod p\,.$$

This shows that $(r, H(m))$ must be a solution of the equation

$$a + b \cdot x + c \cdot y = 0 \mod p \tag{5}$$

where $a := t - \sigma^e \bmod N$, $b := 2^{n_r}$ and $c := 2^8$ are known. This bivariate equation in x, y has a small root $(x_0, y_0) = (r, H(m))$. To solve this equation, one can use a result by Herrmann and May [9] based on Coppersmith's technique for finding small roots of polynomial equations [3].

Coppersmith's technique uses LLL to obtain two polynomials $h_1(x, y)$ and $h_2(x, y)$ such that

$$h_1(x_0, y_0) = h_2(x_0, y_0) = 0$$

holds over the integers. Then one computes the resultant between h_1 and h_2 to recover the common root (x_0, y_0). To that end, one must assume that h_1 and h_2 are algebraically independent. This *ad*

hoc assumption makes the method heuristic; nonetheless it turns out to work quite well in practice. Then, given the root (x_0, y_0) one recovers the randomized encoded message $\mu(m)$ and factors N by GCD.

Assuming that $r < N^\gamma$ and $H(m) \leq N^\delta$, for a balanced RSA modulus one gets the condition:

$$\gamma + \delta \leq \frac{\sqrt{2} - 1}{2} \cong 0.207 \tag{6}$$

This means that for a 1024-bit modulus N, the total size of the unknowns x_0 and y_0 can be at most 212 bits. For ISO/IEC 9796-2 signatures with $k_h = 160$, the unknown r can thus be at most 52 bits long.

2.3 Extension to Several Faults Modulo the Same Factor

[4] shows how to extend the attack to multiple faults, in order to improve the bound on the UMP's size. More precisely, given ℓ faults, one gets a system of equations:

$$a_i + b \cdot x_i + c \cdot y_i = 0 \mod p$$

for $1 \leq i \leq \ell$, where a_i, b and c are known and x_i and y_i are unknown and small. The goal being still to recover p. Note that we can assume that $b = 1$ by multiplying the equations by $b^{-1} \bmod N$.

[4] considers a more general system where instead of known constants b and c, one considers known b_i and c_i. More precisely, we are given ℓ different polynomials

$$f_u(x_u, y_u) = a_u + x_u + c_u y_u \tag{7}$$

where each polynomial f_u has a small root (ξ_u, ν_u) modulo p with $|\xi_u| \leq X$ and $|\nu_u| \leq Y$. Note that, as in the basic case, we re-normalized each polynomial f_u to equate the coefficient of x_u in f_u to one.

[4] shows how to extend Coppersmith's attack to these multiple polynomial equations, thereby obtaining a better bound on the UMP size. Theoretically, given a sufficiently large number of faults, the extended attack could tackle cases where $\gamma + \delta$ is asymptotically close to $\frac{1}{2}$. However, the attack's complexity grows exponentially with the number of faults ℓ, hence aiming at $\gamma + \delta$ values significantly higher than the single fault maximum of 0.207 is totally impractical. We refer the reader to Table 2 illustrating how intractable the problem gets as $\gamma + \delta$ approaches $\frac{1}{2}$.

3 A New Multiple Faults Attack

The previous attack is only applicable for a small number of faults because the lattice dimension grows exponentially with the number of faults. This section describes a different attack that can take advantage of a large number of faults and thus handle much longer UMPs. Indeed, in the new attack, the lattice dimension remains equal to the number of faults, plus one.

As previously, we consider an encoding function given by equation (4)

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

Given a set of faulty signatures σ_i such that:

$$\sigma_i^e = \mu(m_i) = t + r_i \cdot 2^{n_r} + H(m_i) \cdot 2^8 \mod p$$

we get a set of equations of the form

$$A_i + B \cdot x_i + D \cdot y_i = 0 \mod p$$

where $A_i := t - \sigma_i^e \mod N$, $B := 2^{n_r}$ and $D := 2^8$ are known. As in previous section, we can assume that $B = 1$ by multiplying by the equations by $B^{-1} \mod N$. This results in the following equations

$$a_i + x_i + c \cdot y_i = 0 \mod p \tag{8}$$

for $1 \leq i \leq \ell$, where ℓ is the number of faulty signatures. Note that as opposed to equations (7) in the previous section, here we have a constant coefficient c, as in our fault attack.[3]

The new attack is similar to the one in [16]. Applying LLL [18] to the lattice spanned by the columns of the following matrix

$$\begin{pmatrix} \kappa a_1 & \kappa a_2 & \cdots & \kappa a_\ell & \kappa N \\ 1 & 0 & \cdots & 0 & 0 \\ & 1 & \ddots & \vdots & \vdots \\ & & \ddots & 0 & 0 \\ & & & 1 & 0 \end{pmatrix} \tag{9}$$

for a sufficiently large constant κ (as described in [17]), the attacker computes a short vector $(u_1, \ldots, u_\ell)$ such that

$$\sum_{i=1}^{\ell} u_i \cdot a_i = 0 \mod N$$

This implies from (8)

$$\sum_{i=1}^{\ell} u_i \cdot x_i + c \cdot \left(\sum_{i=1}^{\ell} u_i \cdot y_i \right) = 0 \mod p$$

Letting

$$\alpha_0 := \sum_{i=1}^{\ell} u_i \cdot x_i \text{ and } \beta_0 := \sum_{i=1}^{\ell} u_i \cdot y_i \tag{10}$$

this gives:

$$\alpha_0 + c \cdot \beta_0 = 0 \mod p$$

Therefore the vector (α_0, β_0) belongs to the lattice

$$L(c, p) = \left\{ (\alpha, \beta) \in \mathbb{Z}^2 \mid \alpha + c \cdot \beta = 0 \bmod p \right\} \tag{11}$$

We see that if the u_i's are small, then (α_0, β_0) is a short vector in the lattice $L(c, p)$. More precisely, let v be a shortest non-zero vector of $L(c, p)$. If the u_i's are sufficiently small such that $\|(\alpha_0, \beta_0)\| < \|v\|$, then by definition of v we must have $\alpha_0 = \beta_0 = 0$. In this case we get:

$$\sum_{i=1}^{\ell} u_i \cdot x_i = \sum_{i=1}^{\ell} u_i \cdot y_i = 0$$

[3] The attack in this section would not work with different c_i's.

which means that the known vector $(u_1, \ldots, u_\ell)$ is orthogonal (in $\mathbb{Z}$) to the two unknown vectors $(x_1, \ldots, x_\ell)$ and $(y_1, \ldots, y_\ell)$.

Actually, the LLL reduction of lattice (9) yields many other vectors (u_i) which are orthogonal in $\mathbb{Z}$ to both (x_i) and (y_i). Assuming that we can generate $\ell - 2$ such vectors, we can then obtain a bi-dimensional lattice containing both vectors $\boldsymbol{x} = (x_i)$ and $\boldsymbol{y} = (y_i)$. Let $\boldsymbol{x'}$ and $\boldsymbol{y'}$ be a basis of this lattice. Such basis can be obtained by applying LLL a second time to the lattice spanned by the columns of:

$$\begin{pmatrix} \kappa' u_{1,1} & \cdots & \kappa' u_{1,\ell} \\ \vdots & & \vdots \\ \kappa' u_{\ell-2,1} & \cdots & \kappa' u_{\ell-2,\ell} \\ 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}$$

for a sufficiently large constant κ'.

Consider now a vector $\boldsymbol{v} = (v_i)$ that is orthogonal modulo N to both $\boldsymbol{x'}$ and $\boldsymbol{y'}$, that is:

$$\sum_{i=1}^{\ell} v_i \cdot x'_i = 0 \mod N, \qquad \sum_{i=1}^{\ell} v_i \cdot y'_i = 0 \mod N$$

Then since $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the lattice spanned by $\boldsymbol{x'}$ and $\boldsymbol{y'}$, we must have

$$\begin{cases} \boldsymbol{x} = \alpha \cdot \boldsymbol{x'} + \beta \cdot \boldsymbol{y'} \\ \boldsymbol{y} = \alpha' \cdot \boldsymbol{x'} + \beta' \cdot \boldsymbol{y'} \end{cases}$$

for some $\alpha, \alpha', \beta, \beta' \in \mathbb{Z}$. This implies:

$$\sum_{i=1}^{\ell} v_i \cdot x_i = 0 \mod N, \qquad \sum_{i=1}^{\ell} v_i \cdot y_i = 0 \mod N$$

Then from equation (8) this gives:

$$\sum_{i=1}^{\ell} v_i \cdot a_i = 0 \mod p \tag{12}$$

Therefore $\gcd(\sum_i v_i \cdot a_i, N) = p$. Note that the previous computation can be simplified by restricting ourselves to the three first components of $\boldsymbol{x'}$ and $\boldsymbol{y'}$; in that case, one obtains a unique (up to a multiplicative constant) tri-dimensional vector $\boldsymbol{v}$ orthogonal modulo N to both the first three components of $\boldsymbol{x'}$ and the first three components of $\boldsymbol{y'}$. Then equation (12) still holds and as previously $\gcd(\sum_i v_i \cdot a_i, N) = p$.

It remains to justify why we can have $\|(\alpha_0, \beta_0)\| < \|v\|$, where v is a shortest non-zero vector of $L(c, p)$. We provide an argument similar to [20] (see also [8] for higher lattice dimensions). We define a lattice $L(c, p)$ to be B-good if any non-zero vector has a norm $> B$; we say that the lattice is B-bad otherwise. Consider a fixed prime p. By definition of lattice $L(c, p)$ in (11), the value of c modulo p is determined by any non-zero vector in $L(c, p)$. Since there are at most $4B^2$ vectors in the disc of radius B, there are at most $4B^2$ lattices $L(c, p)$ which are B-bad. Therefore for a

random c modulo p, the probability that a lattice is B-bad is at most $4B^2/p$. Taking $B := \sqrt{p}/3$, the probability that a lattice is B-bad is then at most $\frac{1}{2}$.

Conversely a lattice is B-good with probability at least $\frac{1}{2}$. This implies that if $\|(\alpha_0, \beta_0)\| \leq \sqrt{p}/3$, then with probability at least $\frac{1}{2}$ the vector (α_0, β_0) is shorter than the shortest non-zero vector in $L(c, p)$, which implies that $\alpha_0 = \beta_0 = 0$ as required.

Here we have considered a fixed p and a random c modulo p; however in our attack c is a fixed integer and p is random; therefore the previous analysis is heuristic only. More generally, if integers α_0 and β_0 have different sizes, we obtain that $\alpha_0 = \beta_0 = 0$ under the condition:

$$|\alpha_0| \cdot |\beta_0| < \frac{p}{3} \tag{13}$$

Using LLL we expect to obtain vectors (u_i) of norm roughly $N^{1/\ell}$, where ℓ is the number of faulty signatures (see [16]). Let X and Y be upper bounds for the unknowns x_i and y_i. We thus obtain from (10)

$$|\alpha_0| \leq N^{1/\ell} \cdot X, \quad |\beta_0| \leq N^{1/\ell} \cdot Y$$

From (13) we obtain the following bound

$$N^{2/\ell} \cdot X \cdot Y < \frac{p}{3}$$

With $X = N^\gamma$ and $Y = N^\delta$ this yields approximately

$$\frac{2}{\ell} + \gamma + \delta < \frac{1}{2} \tag{14}$$

Therefore, for a large number of faults ℓ we obtain the same asymptotic bound $\gamma + \delta < \frac{1}{2}$ as in [4]; however the lattice dimension in (9) is only $\ell + 1$ instead of being exponential in ℓ as in [4]. In Section 5 we provide the result of practical simulations validating the attack. We then apply the new technique to the EMV specifications in Section 6.

4 Recovering Unknown Moduli

In many practical situations, the modulus N may not be available to the attacker. While contrary to a basic cryptographic assumption, this is frequently the case in proprietary applications. The technique described in the previous section can be adapted to recover *unknown* moduli N from *correct* signatures when the public exponent e is small. Once N has been recovered, one can then apply the fault attack described in the previous section.

As previously, we consider an encoding function given by equation (4)

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

Given a set of ℓ *correct* signatures σ_i such that

$$\sigma_i^e = \mu(m_i) = t + r_i \cdot 2^{n_r} + H(m_i) \cdot 2^8 \bmod N$$

we obtain a set of ℓ equations of the form

$$A_i + B \cdot x_i + D \cdot y_i = 0 \mod N \tag{15}$$

where $A_i := t - \sigma_i^e$, $B := 2^{n_r}$ and $D := 2^8$ are known, but x_i, y_i and N are unknown. Note that as opposed to the previous section A_i is not reduced modulo N; therefore the bit-size of A_i is approximately $e \cdot \log_2 N$.

As in the previous section, using LLL we can find a short vector (u_i) such that

$$\sum_{i=1}^{\ell} u_i \cdot A_i = 0$$

in $\mathbb{Z}$. This implies from (15)

$$B \cdot \left(\sum_{i=1}^{\ell} u_i \cdot x_i \right) + D \cdot \left(\sum_{i=1}^{\ell} u_i \cdot y_i \right) = 0 \mod N.$$

As previously, if the u_i's are sufficiently small, then we will have $\sum_i u_i \cdot x_i = \sum_i u_i \cdot y_i = 0$ over $\mathbb{Z}$. Then again, from $\ell - 2$ linearly independent vectors (u_i) one can recover a 2-dimensional lattice containing the two vectors (x_i) and (y_i).

We proceed by computing two vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ which are both orthogonal in $\mathbb{Z}$ to any vector in this bi-dimensional lattice. This implies that both vectors are orthogonal in $\mathbb{Z}$ to the two vectors (x_i) and (y_i). Equation (15) implies that $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are both orthogonal modulo N to the vector (A_i); therefore to recover N we simply compute the GCD of their respective scalar products with the vector (A_i).

Since the norm of the vector (A_i) is roughly N^e, we can expect (see [16]) to find a vector (u_i) of norm $\cong N^{e/(\ell-1)}$. Moreover, letting $\alpha_0 = \sum_i u_i \cdot x_i$ and $\beta_0 = \sum_i u_i \cdot y_i$, as in the previous section we must have $|\alpha_0| \cdot |\beta_0| < \frac{N}{3}$ so that $\alpha_0 = \beta_0 = 0$ holds with good (heuristic) probability. This gives the following approximate bound

$$N^{2e/(\ell-1)} \cdot X \cdot Y < \frac{N}{3}$$

i.e. using the previous notations

$$\frac{2e}{\ell - 1} + \gamma + \delta < 1 \tag{16}$$

and the required number of signatures:

$$\ell > \frac{2e}{1 - \gamma - \delta} + 1$$

In other words, the number of required signatures is proportional to the public exponent e; this means the modulus recovery technique is practical only for small public exponents. We show in Section 5.2 that it then works well in practice.

5 Simulation Results

5.1 Multiple Fault Attack

We have simulated the fault attack described in Section 3 as follows: We first generate a correct $\sigma_p = \mu(m)^d \mod p$ and a random $\sigma_q \in \mathbb{Z}_q$ and then compute the faulty signature σ using the CRT. This mimics the process described in [4] where concrete faults are injected into devices generating randomized ISO/IEC 9796-2 signatures.

Number of faults ℓ	12	13	14
Success rate with $\gamma = \delta = \frac{1}{6}$	13%	100%	100%
Success rate with $\gamma = \frac{1}{4}$, $\delta = \frac{1}{12}$	0%	100%	100%
Average CPU time (seconds)	0.19	0.14	0.17

Table 1. Attack Simulation Results Using SAGE. Random 1024-bit moduli. Single 2.5 GHz Intel CPU core.

Simulation results are summarized in Table 1. We compute the attack's success rate for $\gamma+\delta = \frac{1}{3}$ for 12, 13 and 14 faults. Theory predicts success with good probability when $\ell > 12$. Table 1 confirms this prediction for both balanced and unbalanced γ and δ configurations

Table 2 provides a comparison with [4]'s multi-fault attack. For large ℓ, the new attack has the asymptotic condition $\gamma + \delta < \frac{1}{2}$, identical to the theoretical asymptotic bound of [4]'s multi-fault variant. It is however considerably easier to deal with cases where $\gamma + \delta$ approaches $\frac{1}{2}$ with the new attack than it is in [4]. Namely, as illustrated in Table 2 when $\gamma + \delta$ approaches $\frac{1}{2}$ [4]'s lattice dimension makes the attack completely impractical. In particular, we show in Section 6 that the new attack allows to attack EMV signature formats that were beyond [4]'s reach.

However we note that for smaller $\gamma + \delta$ values, [4] can be more practical since it requires fewer faulty signatures; for example for $\gamma+\delta = 0.214$ only 2 faulty signatures are required instead of eight in the new attack. In other words, the two techniques nicely complement each other and provide the attacker with a toolbox allowing to adapt his technique to the target's γ and δ configuration.

$\gamma + \delta$	ℓ_{new}	ω_{new}	CPU time (new)	ℓ_{old}	ω_{old}	CPU time (old)
0.204	7	8	0.03 s	3	84	49 s
0.214	8	9	0.04 s	2	126	22 min
0.230	8	9	0.04 s	2	462	—
0.280	10	11	0.07 s	6	6188	—
0.330	14	15	0.17 s	8	2^{21}	—
0.400	25	26	1.44 s	—	—	—
0.450	70	71	36.94 s	—	—	—

Table 2. Comparison of the new attack with [4] for a random 1024-bit modulus.

Explanatory notes regarding Table 2: Table 2 provides, for several values of $\gamma+\delta$, the following information: the number of faulty signatures ℓ_{new} used in our simulation, the corresponding lattice dimension ω_{new}, and the running time of the new attack. For the attack described in [4], the table lists the minimal lattice dimension ω_{old} required to tackle the $\gamma + \delta$ values we consider and the corresponding number of faulty signatures ℓ_{old}. We find ω_{old} by exhaustive search over parameters (ℓ, t, m) with $\ell < 50$, $m < 80$. For $\gamma + \delta = 0.214$ and $\gamma + \delta = 0.23$, one can actually get away with slightly smaller lattice dimensions than indicated in the table (120 and 378 instead of 126 and 462) at the price of more faults (7 and 13 respectively).

5.2 Recovering Unknown Moduli

We have also implemented the technique described in Section 4 to recover N from correct signatures (when N is unknown to the attacker). As shown in Table 3 the attack is quite practical for small

public exponent (e) values. More precisely, we give the success rates for $\gamma+\delta = \frac{1}{3}$ with 10 to 13 valid signatures for $e = 3$. In this case, the theoretical bound (16) predicts that the technique should succeed with good probability when $\ell > 10$; this is well verified for both balanced and unbalanced γ and δ configurations.

Number of signatures ℓ	10	11	12	13
Success rate with $\gamma = \delta = \frac{1}{6}$	2%	59%	61%	61%
Success rate with $\gamma = \frac{1}{4}$, $\delta = \frac{1}{12}$	2%	62%	64%	64%
Average CPU time (seconds)	0.20	0.21	0.25	0.31

Table 3. Modulus recovery simulation in SAGE. Random 1024-bit moduli and $e = 3$. Single core 2.5 GHz Intel CPU.

6 Application to EMV Signatures

6.1 The EMV Specification

EMV is a collection of industry specifications for the inter-operation of payment cards, POS terminals and ATMs. The EMV specification [10] uses ISO/IEC 9796-2 signatures to certify public-keys and to authenticate data. For instance, to authenticate itself, the payment card must issue a signature on data provided by the terminal. The signature algorithm is RSA with ISO/IEC 9796-2 using $e = 3$ or $e = 2^{16} + 1$. The bit length of all moduli is always a multiple of 8. EMV uses special message formats; 7 different formats are used, depending on the message type.

In the following, for clarity's sake, we analyze one of these formats only: the *Offline Dynamic Data Authentication, Dynamic Application Data* format, described in Book 2, Section 6.5, Table 15, page 67 of the EMV specifications [10]. The signing entity is the Card. The message m to be signed has the format $m = m[1] \| m[2]$ with :

$$m[1] = \mathtt{0501}_{16} \parallel \mathsf{L_{DD}} \parallel \mathsf{ICCDD} \parallel \mathtt{BB}_{16} \cdots \mathtt{BB}_{16}$$
$$m[2] = \text{DATA}$$

where $\mathsf{L_{DD}}$ is a byte identifying the length (in bytes) of the ICC Dynamic Data string ICCDD, and DATA is some data provided by the terminal. In general, the ICC Dynamic Data string has the following form:

$$\mathsf{ICCDD} = \mathsf{L_{ICCDN}} \parallel \mathsf{ICCDN} \parallel \mathsf{ADD}$$

where $\mathsf{L_{ICCDN}}$ is one byte identifying the length (in bytes) of the time-variant ICC Dynamic Number ICCDN, and ADD consists of $\mathsf{L_{DD}} - \mathsf{L_{ICCDN}} - 1$ bytes of Additional Dynamic Data to be signed. It is specified that one must have $2 \leq \mathsf{L_{ICCDN}} \leq 8$.

As mentioned in the EMV specifications, the ICC Dynamic Number can be an unpredictable number or a counter incremented for every new signature. In a typical use case (as described, for example, as part of EMV Test 2CC.086.1 Case 07 [11]), ICCDN is a random 8-byte string generated by the card, and ADD is a variable 8-byte string, encoded according to [12]. In this case, we have:

$$m[1] = \mathtt{0501}_{16} \parallel \mathtt{11}_{16} \parallel \mathtt{08}_{16} \parallel \mathsf{ICCDN} \parallel \mathsf{ADD} \parallel \mathtt{BB}_{16} \cdots \mathtt{BB}_{16}$$

which can be rewritten as:

$$m[1] = \mathsf{X} \parallel r \parallel \mathtt{BB}_{16} \cdots \mathtt{BB}_{16}$$

where X is a known value and r is a variable byte string of bit-size $k_r = 128$. This gives:

$$\mu(m) = \mathtt{6A}_{16} \parallel \mathsf{X} \parallel r \parallel \mathtt{BB}_{16} \cdots \mathtt{BB}_{16} \parallel H(m) \parallel \mathtt{BC}_{16} \tag{17}$$

where $H(m)$ is a 160-bit digest of the encoded message m. Note that the no-fault forgery attack from [6] does not apply because here $m[1]$ cannot be controlled by the adversary.

6.2 Fault Attack

The EMV format for $\mu(m)$ given in (17) is the same as the one considered in [4] and recalled in Section 2, and the same as the one considered in our new attack in Section 3. In the particular use case described above, the string X is known but the variables r and $H(m)$ are unknown to the attacker. Therefore the total number of unknown bits is:

$$k_r + k_h = 128 + 160 = 288$$

Hence, for a 1024-bit modulus N, we get:

$$\gamma + \delta = \frac{288}{1024} \approx 0.28$$

which is well beyond the range of practical applicability of [4], as shown in Table 2. However, the new attack will factor N in a fraction of a second using about ten faulty signatures.

References

1. M. Bellare and P. Rogaway, *The Exact security of digital signatures: How to sign with* RSA *and Rabin*, Proceedings of Eurocrypt 1996, LNCS, vol. 1070, Springer-Verlag, 1996, pp. 399–416.
2. D. Boneh, R.A. DeMillo and R.J. Lipton. *On the importance of checking cryptographic protocols for faults*, Journal of Cryptology, 14(2), Springer-Verlag, 2001, pp. 101–119.
3. D. Coppersmith, *Small solutions to polynomial equations, and low exponent vulnerabilities*, Journal of Cryptology, 10(4), 1997, pp. 233–260.
4. J.-S. Coron, A. Joux, I. Kizhvatov, D. Naccache and P. Paillier, *Fault attacks on* RSA *signatures with partially unknown messages*, Proceedings of CHES 2009, LNCS, vol. 5747, Springer-Verlag, 2009, pp. 444–456. Full version: `eprint.iacr.org/2009/309`.
5. J.-S. Coron, D. Naccache and J.P. Stern, *On the security of* RSA *padding*, Proceedings of Crypto 1999, LNCS, vol. 1666, Springer-Verlag, 1999, pp. 1–18.
6. J.-S. Coron, D. Naccache, M. Tibouchi and R. P. Weinmann, *Practical cryptanalysis of* ISO/IEC *9796-2 and* EMV *signatures*, Proceedings of Crypto 2009, LNCS, vol. 5677, Springer-Verlag, 2009. Full version: `eprint.iacr.org/2009/203`.
7. J.-S. Coron, *Optimal security proofs for* PSS *and other signature schemes*, Proceedings of Eurocrypt 2002, LNCS, vol. 2332, Springer-Verlag, 2002, pp. 272–287.
8. J.-S. Coron, M. Joye, D. Naccache and P. Paillier, *Universal padding schemes for* RSA, Proceedings of Crypto 2002, LNCS, vol. 2442, Springer-Verlag, 2002, pp. 226–241.
9. M. Herrmann and A. May, *Solving linear equations modulo divisors: On factoring given any bits*, Proceedings of Asiacrypt 2008, LNCS, vol. 5350, 2008, pp. 406–424.
10. EMV, *Integrated circuit card specifications for payment systems*, Book 2. Security and Key Management. Version 4.2. June 2008. `www.emvco.com`.
11. EMV, *EMVCo type approval terminal level 2 test cases*. Version 4.2a. April 2009. `www.emvco.com`.
12. ISO/IEC 8825-1:2002, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, 2002.
13. ISO/IEC 9796-2, *Information technology – Security techniques – Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash-function*, 1997.

14. ISO/IEC 9796-2:2002 *Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms*, 2002.
15. M. Joye, A. Lenstra, and J.-J. Quisquater, *Chinese remaindering cryptosystems in the presence of faults*, Journal of Cryptology, 21(1), Springer-Verlag, 1999, pp. 27–51.
16. P. Nguyen and J. Stern, *Cryptanalysis of a fast public key cryptosystem presented at SAC '97*, Proceedings of SAC 1998, LNCS, vol. 1556, Springer-Verlag, 1998, pp. 213–218.
17. P. Nguyen and J. Stern, *Merkle-Hellman revisited: a cryptanalysis of the Qu-Vanstone cryptosystem based on group factorization*, Proceedings of Crypto 1997, LNCS, vol. 1294, Springer-Verlag, 1997, pp. 198–212.
18. A. Lenstra, H. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen, vol. 261, Springer-Verlag, 1982, pp. 513–534.
19. W.A. Stein et al., *Sage mathematics software*, Version 4.1, The Sage Development Team, 2009, `www.sagemath.org`.
20. E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, RSA-OAEP *is secure under the* RSA *assumption*, Journal of Cryptology, vol. 17(2), Springer-Verlag, 2004, pp. 81–104.
21. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, vol. 21, ACM, 1978, pp. 120–126.