Fault Attacks on RSA Signatures with Partially Unknown Messages*

Jean-Sébastien Coron[1], Antoine Joux[2], Ilya Kizhvatov[1], David Naccache[3], and Pascal Paillier[4]

[1] Université du Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
{jean-sebastien.coron,ilya.kizhvatov}@uni.lu
[2] DGA and Université de Versailles
UVSQ PRISM 45 avenue des États-Unis, F-78035, Versailles CEDEX, France
antoine.joux@m4x.org
[3] École normale supérieure
Département d'informatique, Groupe de Cryptographie
45, rue d'Ulm, F-75230 Paris CEDEX 05, France
david.naccache@ens.fr
[4] Gemalto, Cryptography & Innovation
6, rue de la Verrerie, F-92447 Meudon sur Seine, France
pascal.paillier@gemalto.com

Abstract. Fault attacks exploit hardware malfunctions to recover secrets from embedded electronic devices. In the late 90's, Boneh, DeMillo and Lipton [6] introduced fault-based attacks on CRT-RSA. These attacks factor the signer's modulus when the message padding function is deterministic. However, the attack does not apply when the message is partially unknown, for example when messages contain some randomness which is recovered only when verifying a *correct* signature.
In this paper we successfully extends RSA fault attacks to a large class of partially known message configurations. The new attacks rely on Coppersmith's algorithm for finding small roots of multivariate polynomial equations. We illustrate the approach by successfully attacking several randomized versions of the ISO/IEC 9796-2 encoding standard. Practical experiments show that a 2048-bit modulus can be factored in less than a minute given one faulty signature containing 160 random bits and an unknown 160-bit message digest.

Keywords: Fault attacks, digital signatures, RSA, Coppersmith's theorem, ISO/IEC 9796-2.

1 Introduction

1.1 Background

RSA [21] is undoubtedly the most common digital signature scheme used in embedded security tokens. To sign a message m with RSA, the signer applies an encoding (padding) function μ to m, and then computes the signature $\sigma = \mu(m)^d \bmod N$. To verify the signature, the receiver checks that

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

As shown by Boneh, DeMillo and Lipton [6] and others (*e.g.* [17]), RSA implementations can be vulnerable to fault attacks, especially when the Chinese Remainder Theorem (CRT) is used; in this case the device computes

$$\sigma_p = \mu(m)^d \mod p\,, \qquad \sigma_q = \mu(m)^d \mod q$$

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

* An extended abstract of this paper will appear at CHES 2009. This is the full version.

Assuming that the attacker is able to induce a fault when σ_q is computed while keeping the computation of σ_p correct, one gets

$$\sigma_p = \mu(m)^d \mod p\,, \qquad \sigma_q \neq \mu(m)^d \mod q$$

and the resulting (faulty) signature σ satisfies

$$\sigma^e = \mu(m) \mod p\,, \qquad \sigma^e \neq \mu(m) \mod q\,.$$

Therefore, given one faulty σ, the attacker can factor N by computing

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

Boneh *et al.*'s fault attack is easily extended to any deterministic RSA encoding, *e.g.* the Full Domain Hash (FDH) [5] 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, *e.g.* as in the Probabilistic Full Domain Hash (PFDH) encoding [10] where the signature is $\sigma \| r$ with $\sigma = H(m \,\|\, r)^d \bmod N$. In that case, given the faulty value of σ and knowing r, the attacker can still factor N by computing

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

1.2 Partially-Known Messages: The Fault-Attacker's Deadlock

However, if the message is not entirely given to the opponent the attack is thwarted, *e.g.* this may occur when the signature has the form $\sigma = (m\|r)^d \bmod N$ where r is a random nonce. Here the verifier can recover r only after completing the verification process; however r can 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

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

In other words, the attacker faces an apparent deadlock: recovering the r used in the faulty signature σ seems to require that σ is a correctly verifiable signature. Yet, obviously, a correct signature does not factor N. These conflicting constraints cannot be conciliated unless r is short enough to be guessed by exhaustive search. Inducing faults in many signatures does not help either since different r values are used in successive signatures (even if m remains invariant). As a result, randomized RSA encoding schemes are usually considered to be inherently immune against fault attacks.

1.3 The New Result

We overcome this apparent deadlock by showing how to extract *in some cases* the unknown message part (UMP) involved in the generation of faulty RSA signatures. We develop several techniques that extend Boneh *et al.*'s attack to a large class of partially known message configurations. We nonetheless assume that certain conditions on the unknown parts of the encoded message are met; these conditions may depend on the encoding function itself and on the hash functions used. To illustrate our attacks, we have chosen to consider the ISO/IEC 9796-2 standard [15]. ISO/IEC 9796-2 is originally a deterministic

encoding scheme often used in combination with message randomization (*e.g.* in EMV [12]). The encoded message has the form:

$$\mu(m) = \mathtt{6A_{16}} \,\|\, m[1] \,\|\, H(m) \,\|\, \mathtt{BC_{16}}$$

where $m = m[1] \,\|\, m[2]$ is split into two parts. We 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 [6][1]. The new method is based on a result by Herrmann and May [11] for finding small roots of linear equations modulo an unknown factor p of N; [11] is itself based on Coppersmith's technique [7] for finding small roots of polynomial equations using the LLL algorithm [19]. We also show how to extend our attack to multiple UMPs and to *scenarii* where more faulty signatures can be obtained from the device.

It is trivially seen that other *deterministic* signature encoding functions such as PKCS#1 v1.5 can be broken by the new attack *even when the message digest is unknown*. We elaborate on this in further detail at the end of the paper.

1.4 The ISO/IEC 9796-2 Standard

ISO/IEC 9796-2 is an encoding standard allowing partial or total message recovery [15, 16]. The encoding can be used with hash functions $H(m)$ of diverse digest sizes k_h. For the sake of simplicity we assume that 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. Therefore the size of $\mu(m)$ is always $k - 1$ bits. Note that the original version of the standard recommended $128 \leq k_h \leq 160$ for partial message recovery (see [15], §5, note 4). In [8], Coron, Naccache and Stern introduced an attack against ISO/IEC 9796-2; the authors estimated that attacking $k_h = 128$ and $k_h = 160$ would require respectively 2^{54} and 2^{61} operations. After Coron *et al.*'s publication, ISO/IEC 9796-2 was amended and the current official requirement (see [16]) is now $k_h \geq 160$. In a recent work Coron, Naccache, Tibouchi and Weinmann successfully attack the currently valid version of ISO/IEC 9796-2 [9].

To illustrate our purpose, we consider a message $m = m[1] \,\|\, m[2]$ of the form

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

where r is a message part unknown to the adversary, α and α' are strings known to the adversary and DATA is some known or unknown string[2]. The size of r is denoted 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 the total number of unknown bits in $\mu(m)$ is $k_r + k_h$.

[1] In our attack, it does not matter how large the unknown part of $m[2]$ is.
[2] The attack will work equally well in both cases.

2 Fault Attack on Partially-Known Message ISO/IEC 9796-2

This section extends [6] to signatures of partially known messages encoded as described previously. We assume that after injecting a fault the opponent is in possession of a faulty signature σ such that:

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

From (2) we can write

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

where t is a known value. Note that both r and $H(m)$ are unknown to the adversary. From (3) we obtain:

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

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. Therefore we are left with solving equation (5) which is linear in the two variables x, y and admits a small root $(x_0, y_0) = (r, H(m))$. However the equation holds modulo an unknown divisor p of N and not modulo N. Such equations were already exploited by Herrmann and May [11] to factor an RSA modulus $N = pq$ when some blocks of p are known. Their method is based on Coppersmith's technique for finding small roots of polynomial equations [7]. 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, we 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.

Theorem 1 (Herrmann-May [11]). *Let N be a sufficiently large composite integer with a divisor $p \geq N^\beta$. Let $f(x, y) = a + b \cdot x + c \cdot y \in \mathbb{Z}[x, y]$ be a bivariate linear polynomial. Assume that $f(x_0, y_0) = 0 \bmod p$ for some (x_0, y_0) such that $|x_0| \leq N^\gamma$ and $|y_0| \leq N^\delta$. Then for any $\varepsilon > 0$, under the condition*

$$\gamma + \delta \leq 3\beta - 2 + 2(1 - \beta)^{3/2} - \varepsilon \tag{6}$$

one can find $h_1(x, y), h_2(x, y) \in \mathbb{Z}[x, y]$ such that $h_1(x_0, y_0) = h_2(x_0, y_0) = 0$ over $\mathbb{Z}$, in time polynomial in $\log N$ and ε^{-1}.

We only sketch the proof and refer the reader to [11] for more details. Assume that $b = 1$ in the polynomial f (otherwise multiply f by $b^{-1} \bmod N$) and consider the polynomial

$$f(x, y) = a + x + c \cdot y$$

We look for (x_0, y_0) such that $f(x_0, y_0) = 0 \bmod p$. The basic idea consists in generating a family $\mathcal{G}$ of polynomials admitting (x_0, y_0) as a root modulo p^t for some large enough integer t. Any linear combination of these polynomials will also be a polynomial admitting (x_0, y_0) as a root modulo p^t. We will use LLL to find such polynomials with small coefficients. To do so, we view any polynomial

$h(x,y) = \sum h_{i,j} x^i y^j$ as the vector of coefficients $\left(h_{i,j} X^i Y^j\right)_{i,j}$ and denote by $\|h(xX, yY)\|$ this vector's Euclidean norm. Performing linear combinations on polynomials is equivalent to performing linear operations on their vectorial representation, so that applying LLL to the lattice spanned by the vectors in $\mathcal{G}$ will provide short vectors representing polynomials with root (x_0, y_0) mod p^t.

We now define the family $\mathcal{G}$ of polynomials as

$$g_{k,i}(x,y) := y^i \cdot f^k(x,y) \cdot N^{\max(t-k,0)}$$

for $0 \leq k \leq m$, $0 \leq i \leq m-k$ and integer parameters t and m. For all values of indices k, i, it holds that

$$g_{k,i}(x_0, y_0) = 0 \mod p^t.$$

We first sort the polynomials $g_{k,i}$ by increasing k values and then by increasing i values. Denoting $X = N^\gamma$ and $Y = N^\delta$, we write the coefficients of the polynomial $g_{k,i}(xX, yY)$ in the basis $x^{k'} \cdot y^{i'}$ for $0 \leq k' \leq m$ and $0 \leq i' \leq m - k'$. This results in the matrix of row vectors illustrated in Figure 1. Let

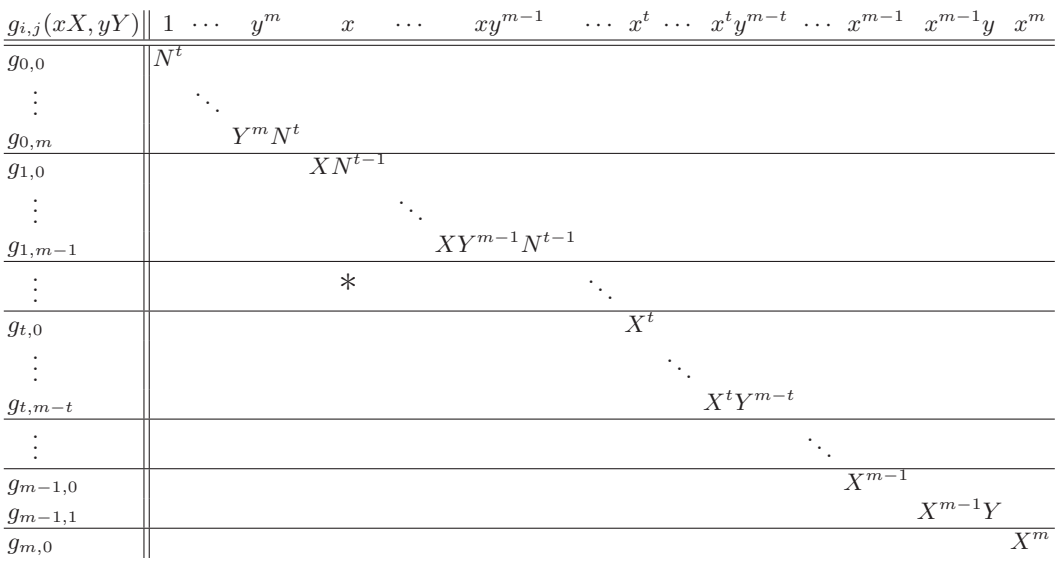

Fig. 1. Lattice of row vectors formed by the coefficients of the polynomials $g_{k,i}(xX, yY)$. The matrix is lower triangular; we only represent the diagonal elements.

L be the corresponding lattice; L's dimension is

$$\omega = \dim(L) = \frac{m^2 + 3m + 2}{2} = \frac{(m+1)(m+2)}{2}$$

and we have

$$\det L = X^{s_x} Y^{s_y} N^{s_N}$$

where

$$s_x = s_y = \sum_{k=0}^{m} \sum_{i=0}^{m-k} i = \frac{m(m+1)(m+2)}{6}$$

and

$$s_N = \sum_{i=0}^{t}(m+1-i)(t-i) = \frac{t}{6}(t+1)(t-3m-4).$$

We now apply LLL to the lattice L to find two polynomials $h_1(x,y)$ and $h_2(x,y)$ with small coefficients.

Theorem 2 (LLL [19]). *Let L be a lattice spanned by $(u_1,\ldots,u_\omega)$. Given the vectors $(u_1,\ldots,u_\omega)$, the* LLL *algorithm finds in polynomial time two linearly independent vectors b_1, b_2 such that*

$$\|b_1\|, \|b_2\| \leq 2^{\omega/4}(\det L)^{1/(\omega-1)} .$$

Therefore using LLL we can get two polynomials $h_1(x,y)$ and $h_2(x,y)$ such that

$$\|h_1(xX,yY)\|, \|h_2(xX,yY)\| \leq 2^{\omega/4} \cdot (\det L)^{1/(\omega-1)} . \tag{7}$$

Using Howgrave-Graham's lemma (below), we can determine the required bound on the norms of h_1 and h_2 to ensure that (x_0,y_0) is a root of both h_1 and h_2 over the integers:

Lemma 1 (Howgrave-Graham [13]). *Assume that $h(x,y) \in \mathbb{Z}[x,y]$ is a sum of at most ω monomials and assume further that $h(x_0,y_0) = 0 \mod B$ where $|x_0| \leq X$ and $|y_0| \leq Y$ and $\|h(xX,yY)\| < B/\sqrt{\omega}$. Then $h(x_0,y_0) = 0$ holds over the integers.*

Proof. We have

$$\begin{aligned}
|h(x_0,y_0)| &= \left|\sum h_{ij}x_0^i y_0^i\right| = \left|\sum h_{ij}X^iY^j\left(\frac{x_0}{X}\right)^i\left(\frac{y_0}{Y}\right)^j\right| \\
&\leq \sum\left|h_{ij}X^iY^j\left(\frac{x_0}{X}\right)^i\left(\frac{y_0}{Y}\right)^j\right| \leq \sum\left|h_{ij}X^iY^j\right| \\
&\leq \sqrt{\omega}\|h(xX,yY)\| < B
\end{aligned}$$

Since $h(x_0,y_0) = 0 \bmod B$, this implies that $h(x_0,y_0) = 0$ over the integers. □

We apply Lemma 1 with $B := p^t$. Using (7) this gives the condition:

$$2^{\omega/4} \cdot (\det L)^{1/(\omega-1)} \leq \frac{N^{\beta t}}{\sqrt{\omega}} . \tag{8}$$

[11] shows that by letting $t = \tau \cdot m$ with $\tau = 1 - \sqrt{1-\beta}$, we get the condition:

$$\gamma + \delta \leq 3\beta - 2 + 2(1-\beta)^{3/2} - \frac{3\beta(1+\sqrt{1-\beta})}{m}$$

Therefore we obtain as in [11] the following condition for m:

$$m \geq \frac{3\beta(1+\sqrt{1-\beta})}{\varepsilon} .$$

Since LLL runs in time polynomial in the lattice's dimension and coefficients, the running time is polynomial in $\log N$ and $1/\varepsilon$.

2.1 Discussion

For a balanced RSA modulus ($\beta = 1/2$) we get the condition:

$$\gamma + \delta \leq \frac{\sqrt{2}-1}{2} \cong 0.207 \tag{9}$$

This means that for a 1024-bit RSA modulus N, the total size of the unknowns x_0 and y_0 can be at most 212 bits. Applied to our context, this implies that for ISO/IEC 9796-2 with $k_h = 160$, the size of the UMP r can be as large as 52 bits. Section 3 reports practical experiments confirming this prediction. In Appendix A we provide a Python code for computing the bound on the size of the unknown values $(k_r + k_h)$ as a function of the modulus size.

In the next paragraph we extend the method to 1) several disjoint UMP blocks in the encoding function, 2) to two dissimilar faults (one modulo p and one modulo q) and 3) to two or more faults modulo the same prime factor.

2.2 Extension to Several Unknown Bits Blocks

Assume that the UMP used in ISO/IEC 9796-2 is split into n different blocks, namely

$$\mu(m) = \mathtt{6A_{16}} \,\|\, \alpha_1 \,\|\, r_1 \,\|\, \alpha_2 \,\|\, r_2 \,\|\, \cdots \,\|\, \alpha_n \,\|\, r_n \,\|\, \alpha_{n+1} \,\|\, H(m) \,\|\, \mathtt{BC_{16}} \tag{10}$$

where the UMPs $r_1, \ldots, r_n$ are all part of the message m. The α_i blocks are known. We use the extended result of Herrmann and May [11], allowing to (heuristically) find the solutions $(y_1, \ldots, y_n)$ of a linear equation modulo a factor p of N:

$$a_0 + \sum_{i=1}^{n} a_i \cdot x_i = 0 \mod p$$

with $p \geq N^{\beta}$ and $|y_i| \leq N^{\gamma_i}$, under the condition [11]

$$\sum_{i=1}^{n} \gamma_i \leq 1 - (1-\beta)^{\frac{n+1}{n}} - (n+1)(1 - \sqrt[n]{1-\beta})(1-\beta) \; .$$

For $\beta = 1/2$ and for a large number of blocks n, one gets the bound

$$\sum_{i=1}^{n} \gamma_i \leq \frac{1 - \ln 2}{2} \cong 0.153$$

This shows that if the total number of unknown bits plus the message digest is less than 15.3% of the size of N, then the UMPs can be fully recovered from the faulty signature and Boneh *et al.*'s attack will apply again. However the number of blocks cannot be too large because the attack's runtime increases exponentially with n.

2.3 Extension to Two Faults Modulo Different Factors

Assume that we can get two faulty signatures, one incorrect modulo p and the other incorrect modulo q. This gives the two equations

$$\begin{aligned} a_0 + b_0 \cdot x_0 + c_0 \cdot y_0 = 0 \mod p \\ a_1 + b_1 \cdot x_1 + c_1 \cdot y_1 = 0 \mod q \end{aligned}$$

with small unknowns x_0, y_0, x_1, y_1. By multiplying the two equations, we get the quadri-variate equation:

$$a_0a_1 + a_0b_1 \cdot x_1 + b_0a_1 \cdot x_0 + a_0c_1 \cdot y_0 + c_0a_1 \cdot y_1 + b_0b_1 \cdot x_0x_1 + b_0c_1 \cdot x_0y_1 + c_0b_1 \cdot y_0x_1 + c_0c_1 \cdot y_0y_1 = 0 \mod N$$

We can linearize this equation (replacing products of unknowns with new variables) and obtain a new equation of the form:

$$\alpha_0 + \sum_{i=1}^{8} \alpha_i \cdot z_i = 0 \mod N$$

where the coefficients $\alpha_0, \ldots, \alpha_8$ are known and the unknowns $z_1, \ldots, z_8$ are small. Using LLL again, we can recover the z_i's (and then x_0, x_1, y_0, y_1) as long as the cumulated size of the z_i's is at most the size of N. This yields the condition

$$6 \cdot (k_r + k_h) \leq k$$

which, using the notation of Theorem 1, can be reformulated as

$$\gamma + \delta \leq \frac{1}{6} \cong 0.167 \, .$$

This remains weaker than condition (9). However the attack is significantly faster because it works over a lattice of constant dimension 9. Moreover, the 16.7% bound is likely to lend itself to further improvements using Coppersmith's technique instead of plain linearization.

2.4 Extension to Several Faults Modulo the Same Factor

To exploit single faults, we have shown how to use lattice-based techniques to recover p given N and a bivariate linear equation $f(x, y)$ admitting a small root (x_0, y_0) modulo p. In this context, we have used Theorem 1 which is based on approximate GCD techniques from [14]. In the present section we would like to generalize this to use ℓ different polynomials of the same form, each having a small root modulo p. More precisely, let ℓ be a fixed parameter and assume that as the result of ℓ successive faults, we are given ℓ different polynomials

$$f_u(x_u, y_u) = a_u + x_u + c_u y_u \tag{11}$$

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 ensure that the coefficient of x_u in f_u is equal to one. To avoid double subscripts, we hereafter use the Greek letters ξ and ν to represent the root values. We would like to use a lattice approach to construct new multivariate polynomials in the variables $(x_1, \cdots, x_\ell, y_1, \cdots, y_\ell)$ with the root $R = (\xi_1, \cdots, \xi_\ell, \nu_1, \cdots, \nu_\ell)$. To that end we fix two

parameters m and t and build a lattice on a family of polynomials $\mathcal{G}$ of degree at most m with root R modulo $B = p^t$. This family is composed of all polynomials of the form

$$y_1^{i_1}\, y_2^{i_2} \cdots y_\ell^{i_\ell}\, f_1(x_1, y_1)^{j_1}\, f_2(x_2, y_2)^{j_2} \cdots f_\ell(x_\ell, y_\ell)^{j_\ell}\, N^{\max(t-j,0)} \ ,$$

where each i_u, j_u is non-negative, $i = \sum_{u=1}^{\ell} i_u$, $j = \sum_{u=1}^{\ell} j_u$ and $0 \leq i + j \leq m$. Once again, let L be the corresponding lattice. Its dimension ω is equal to the number of monomials of degree at most m in 2ℓ unknowns, *i.e.*

$$\omega = \binom{m + 2\ell}{2\ell} \ .$$

Since we have a common upper bound X for all values $|\xi_u|$ and a common bound for all $|\nu_u|$ we can compute the lattice's determinant as

$$\det(L) = X^{s_x} Y^{s_y} N^{s_N} \ ,$$

where s_x is the sum of the exponents of all unknowns x_u in all occurring monomials, s_y is the sum of the exponents of the y_u and s_N is the sum of the exponents of N in all occurring polynomials. For obvious symmetry reasons, we have $s_x = s_y$ and noting that the number of polynomials of degree exactly d in ℓ unknowns is $\binom{d+\ell-1}{\ell-1}$ we find

$$s_x = s_y = \sum_{d=0}^{m} d \binom{d + \ell - 1}{\ell - 1} \binom{m - d + \ell}{\ell} = \frac{\ell(2\ell + m)!}{(2\ell + 1)!(m - 1)!}.$$

Likewise, summing on polynomials with a non-zero exponent v for N, where the sum of the j_u is $t - v$ we obtain

$$\begin{aligned} s_N &= \sum_{v=1}^{t} v \binom{t - v + \ell - 1}{\ell - 1} \binom{m - t + v + \ell}{\ell} \\ &= \binom{1 + \ell + m - t}{\ell} \binom{-2 + \ell + t}{\ell - 1}\, {}_3F_2 \left[\begin{matrix} 2 & & 2 - \ell - t \\ 1 - t & ; & 2 + m - t \\ 2 + \ell + m - t & & \end{matrix} ; 1 \right] \end{aligned}$$

where ${}_3F_2$ is the generalized hypergeometric function.

As usual, assuming that $p = N^\beta$ we can find a polynomial with the correct root over the integers under the condition of formula (8).

Concrete Bounds: Using the notation of Theorem 1, we compute effective bounds on $\gamma + \delta = \log(XY)/\log(N)$ from the logarithm of condition (8), dropping the terms $\sqrt{\omega}$ and $2^{\omega/4}$ which become negligible as N grows. For concrete values of N, bounds are slightly smaller. Dividing by $\log(N)$, we find

$$s_x \cdot (\gamma + \delta) + s_N \leq \beta t \omega \ .$$

Thus, given k, t and m, we can achieve at best

$$\gamma + \delta \leq \frac{\beta t \omega - s_N}{s_x}.$$

We have computed the achievable values of $\gamma + \delta$ for $\beta = 1/2$, for various parameters and for lattice dimensions $10 \leq \omega \leq 1001$. Results are given in Table 4, Appendix B.

Recovering the Root: With 2ℓ unknowns instead of two, applying usual heuristics and hoping that lattice reduction directly outputs 2ℓ algebraically independent polynomials with the prescribed root over the integers becomes a wishful hope. Luckily, a milder heuristic assumption suffices to make the attack work. The idea is to start with K equations instead of ℓ and iterate the lattice reduction attack for several subsets of ℓ equations chosen amongst the K available equations. Potentially, we can perform $\binom{K}{\ell}$ such lattice reductions. Clearly, since each equation involves a different subset of unknowns, they are all different. Note that this does not suffice to guarantee algebraic independence; in particular, if we generate more than K equations they cannot be algebraically independent. However, we only need to ascertain that the root R can be extracted from the available set of equations. This can be done, using Gröbner basis techniques, under the heuristic assumption that the set of equations spans a multivariate ideal of dimension zero *i.e.* that the number of solutions is finite.

Note that we need to choose reasonably small values of ℓ and K to be able to use this approach in practice. Indeed, the lattice that we consider should not become too large and, in addition, it should be possible to solve the resulting system of equations using either resultants or Buchberger's algorithm which means that neither the degree nor the number of unknowns should increase too much.

Asymptotic Bounds: Despite the fact that we cannot hope to run the multi-polynomial variant of our attack when parameters become too large, it is interesting to determine the theoretical limit of the achievable value of $\gamma+\delta$ as the number of faults ℓ increases. To that end, we assume as previously that $\beta = 1/2$, let $t = \tau m$ and replace ω, s_x and s_N by the following approximations:

$$\omega \cong \frac{m^{2\ell}}{(2\ell)!}\ , \quad s_x \cong \sum_{d=0}^{m} \frac{d^\ell\,(m-d)^\ell}{(\ell-1)!\,\ell!}\ , \quad s_N \cong \sum_{v=1}^{t} v\,\frac{(t-v)^{\ell-1}(m-t+v)^\ell}{(\ell-1)!\,\ell!}\ .$$

For small ℓ values we provide in Table 1 the corresponding bounds on $\gamma+\delta$. Although we do not provide further details here due to lack of space, one can show that the bound $\gamma+\delta$ tends to $1/2$ as the number of faults ℓ tends to infinity and that all $\gamma+\delta$ values are algebraic numbers.

ℓ	1	2	3	4	5	6	7	8	9	10
$\gamma+\delta$	0.207	0.293	0.332	0.356	0.371	0.383	0.391	0.399	0.405	0.410

Table 1. Bound for the relative size $\gamma+\delta$ of the unknowns as a function of the number of faults ℓ.

3 Simulation Results

Assuming that fault injection can be performed on unprotected devices (see Section 4), we simulated the attack. In the experiment we generated faulty signatures (using the factors p and q) and applied to them the attack's mathematical analysis developed in the previous sections to factor N. We refer the reader to Section 4 for more on physical fault injection experiments.

3.1 Single-Fault Attack Simulations

We first consider a single-UMP, single-fault attack when H = SHA-1 *i.e.* $k_h = 160$. Using the SAGE library LLL implementation, computations were executed on a 2GHz Intel notebook.

modulus size k	UMP **size** k_r	m	t	**lattice dimension** ω	**runtime**
1024	6	10	3	66	4 minutes
1024	13	13	4	105	51 minutes
1536	70	8	2	45	39 seconds
1536	90	10	3	66	9 minutes
2048	158	8	2	45	55 seconds

Table 2. Single fault, single UMP 160-bit digests ($k_h = 160$). LLL runtime for different parameter combinations.

Experimental results are summarized in Table 2. We see that for 1024-bit RSA, the randomizer size k_r must be quite small and the attack is less efficient than exhaustive search[3]. However for larger moduli, the attack becomes more efficient. Typically, using a single fault and a 158-bit UMP, a 2048-bit RSA modulus was factored in less than a minute.

3.2 Multiple-Fault Simulations

To test the practicality of the approach presented in Section 2.4, we have set $(\ell, t, m) = (3, 1, 3)$ *i.e.* three faulty signatures. This leads to a lattice of dimension 84 and a bound $\gamma + \delta \leq 0.204$. Experiments were carried out with 1024, 1536 and 2048 bit RSA moduli. This implementation also relied on the SAGE library [20] running on a single PC. Quite surprisingly, we observed a very large number of polynomials with the expected root over the integers. The test was run for three random instances corresponding to the parameters in Table 3.

modulus size k	UMP **size** k_r	**runtime**
1024	40	49 seconds
1536	150	74 seconds
2048	250	111 seconds

Table 3. Three faults, single UMP, 160-bit digests ($k_h = 160$). LLL runtime for different parameter combinations.

For each parameter set, the first 71 vectors in the reduced lattice have the expected root. In fact, it is even possible to solve the system of equations without resorting to Buchberger's algorithm. Instead, we use a much simpler strategy. We first consider the system modulo a prime p' above 2^{160} (or above 2^{250} in the 2048-bit experiment). With this system, linearization suffices to obtain through echelonization the polynomials $x_i - \xi_i$ and $y_i - \nu_i$. Since p' is larger than the bounds on the values, this yields the exact values of ξ_1, ξ_2, ξ_3, ν_1, ν_2 and ν_3. Once this is done, the factors of N are easily recovered by computing the GCD of N with any of the values $f_i(\xi_i, \nu_i)$.

Three faults turn-out to be more efficient than single-fault attacks (Table 3 *vs.* Table 2). In particular for a 1024-bit RSA modulus, the three-fault attack recovered a 40-bit UMP r in 49 seconds[4], whereas the single-fault attack only recovered a 13-bit UMP in 51 minutes.

[3] Exhausting a 13-bit randomizer took 0.13 seconds.
[4] We estimate that exhaustive search on a 40-bit UMP would take roughly a year on the same single PC.

4 Physical Fault Injection Experiments

We performed fault injection on an unprotected device to demonstrate the entire attack flow. We obtain a faulty signature from a general-purpose 8-bit microcontroller running an RSA implementation and factor N using the mathematical attack of Section 2.

Our target device was an Atmel ATmega128 [3], a very pupular RISC microcontroller (μC) with an 8-bit AVR core. The μC was running an RSA-CRT implementation developed in C using the BigDigits multiple-precision arithmetic library [4]. The μC was clocked at 7.3728 MHz using a quartz crystal and powered from a 5V source.

We induced faults using voltage spikes (*cf.* to [1] and [2] for more information on such attacks on similar μCs). Namely, we caused brief power cut-offs (spikes) by grounding the chip's V_{cc} input for short time periods. Spikes were produced by an FPGA-based board counting the μC's clock transitions and generating the spike at a precise moment. The cut-off duration was variable with 10ns granularity and the spike's temporal position could be fine-tuned with the same granularity. The fault was heuristically positioned to obtain the stable fault injection in one of the RSA-CRT branches (computing σ_p or σ_q). A 40ns spike is presented in Figure 2. Longer spike durations caused a μC's reset.

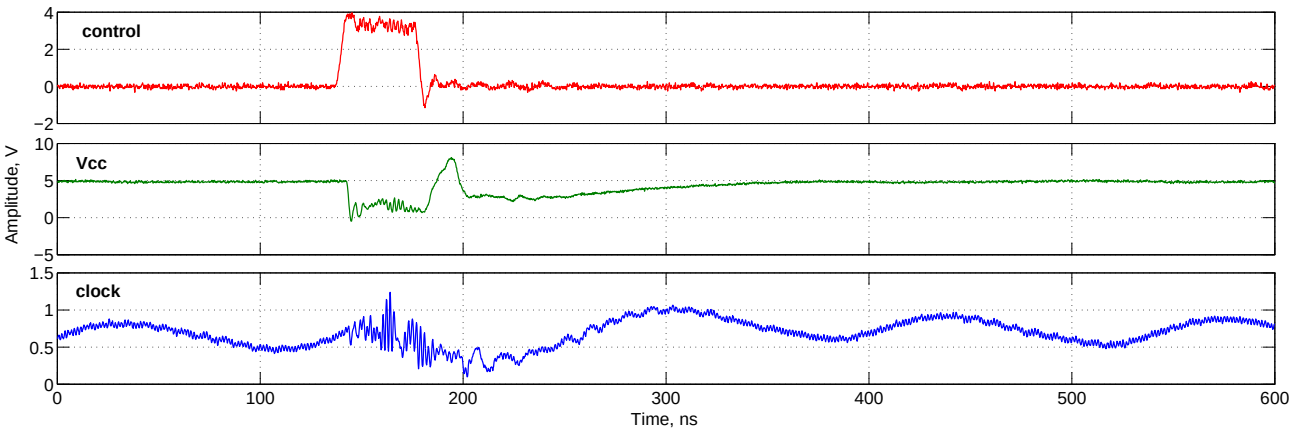

Fig. 2. Spike captured with a DSO: control signal from FPGA, power supply cut-off, and induced glitch in the clock signal.

5 Conclusion

The paper introduced a new breed of partially-known message fault attacks against RSA signatures. These attacks allow to factor the modulus N given a single faulty signature. Although the attack is heuristic, it works well in practice and paradoxically becomes more efficient as the modulus size increases. As several faulty signatures are given longer UMPs and longer digests become vulnerable.

The new techniques are more generally applicable to any context where the signed messages are partially unknown, in which case we provide explicit size conditions for the fault attack to apply. This has a direct impact on other encoding functions, such as PKCS#1 v1.5 standard where a message m is encoded as

$$\mu(m) = \mathtt{0001}_{16} \,\|\, \underbrace{\mathtt{FF}_{16} \ldots \mathtt{FF}_{16}}_{k_1 \text{ bytes}} \,\|\, \mathtt{00}_{16} \,\|\, T \,\|\, H(m)$$

where T is a known sequence of bytes which encodes the identifier of the hash function and k_1 is a size parameter which is adjusted to make $\mu(m)$ have the same number of bytes than the modulus. With a single unknown bounded by N^δ the condition is $\delta < 0.25$. Therefore assuming a 2048-bit modulus and $H =$ SHA-512, we obtain that the modulus can be efficiently factored using a single faulty signature σ even when the signed message is *totally unknown*. This enables fault attacks in complex cryptographic *scenarii* where *e.g.* a smart-card and a terminal exchanging RSA signatures on encrypted messages.

References

1. J.-M. Schmidt and C. Herbst, *A practical fault attack on square and multiply*, Proceedings of FDTC 2008, IEEE Computer Society, 2008, pp. 53–58.
2. C.H. Kim and J.-J. Quisquater, *Fault attacks for* CRT *based* RSA*: new attacks, new results, and new countermeasures*, Proceedings of WISTP 2007, LNCS, vol. 4462, Springer-Verlag, 2007, pp. 215–228.
3. ATmega128 datasheet, `www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.
4. BigDigits multiple-precision arithmetic source code, Version 2.2. `www.di-mgt.com.au/bigdigits.html`.
5. 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.
6. D. Boneh, R.A. DeMillo and R.J. Lipton. *On the importance of checking cryptographic protocols for faults*, Journal of Cryptology, Springer-Verlag, 14(2), pp. 101–119, 2001.
7. D. Coppersmith, *Small solutions to polynomial equations, and low exponent vulnerabilities*. Journal of Cryptology, 10(4), 1997, pp. 233–260.
8. 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.
9. 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, Springer-Verlag, 2009, to appear. Full version: `eprint.iacr.org/2009/203.pdf`
10. J.S. Coron, *Optimal security proofs for* PSS *and other signature schemes*, Proceedings of Eurocrypt'02, LNCS, vol. 2332, Springer-Verlag, 2002, pp. 272–287.
11. 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.
12. EMV *Integrated circuit card specifications for payment systems*, Book 2. Security and Key Management. Version 4.2. June 2008. `www.emvco.com`.
13. N.A. Howgrave-Graham, *Finding small roots of univariate modular equations revisited*. In Cryptography and Coding, LNCS, vol. 1355, Springer Verlag, 1997, pp. 131–142.
14. N.A. Howgrave-Graham, *Approximate integer common divisors*. In CALC, pp. 51–66. 2001.
15. ISO/IEC 9796-2, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash-function*, 1997.
16. ISO/IEC 9796-2:2002 *Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms*, 2002.
17. M. Joye, A. Lenstra, and J-J.Quisquater , *Chinese remaindering cryptosystems in the presence of faults*, Journal of Cryptology, 21(1), 1999, 27–51.
18. B. Kaliski, PKCS*#1:* RSA *Encryption Standard, Version 1.5*, RSA Laboratories, November 1993.
19. A. Lenstra, H. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*. Mathematische Annalen, vol. 261, 1982, pp. 513–534
20. Sage, Mathematical Library. `www.sagemath.org`
21. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, vol. 21, 1978, pp. 120–126.

A Root Size Estimation

```
from math import log,sqrt

def LatticeExpo(t,m):
  sy=sx=sn=w=0
  for k in range(m+1):
    for i in range(m-k+1):
      j=max(t-k,0)
      sy+=i; sx+=k; sn+=j; w+=1
  return (sx,sy,sn,w)

def bound(t,m,n):
  (sx,sy,sn,w)=LatticeExpo(t,m)
  nxy=(w*(n*t*.5-.25*w-log(sqrt(w),2))-n*sn)/sx
  return nxy,w
```

B Achievable Bound on $\gamma + \delta$

We provide in Table 4 the achievable bound on $\gamma + \delta$, as a function of the number of faults ℓ and parameters (t, m).

C Example of a Faulty Signature

Table 5 presents an example of a faulty 1536-bit RSA-CRT signature obtained during our fault injection experiments. Values known to the opponent are underlined.

Dimension ω	Bound $\gamma+\delta$	(ℓ,t,m)	Dimension ω	Bound $\gamma+\delta$	(ℓ,t,m)	Dimension ω	Bound $\gamma+\delta$	(ℓ,t,m)
10	0.100	$(1,1,3)$	15	0.125	$(1,1,4)$	21	0.129	$(1,1,5)$
28	0.143	$(1,2,6)$	36	0.155	$(1,2,7)$	45	0.158	$(1,2,8)$
55	0.161	$(1,3,9)$	66	0.168	$(1,3,10)$	78	0.171	$(1,3,11)$
91	0.172	$(1,3,12)$	105	0.176	$(1,4,13)$	120	0.179	$(1,4,14)$
120	0.179	$(1,4,15)$	153	0.181	$(1,5,16)$	171	0.183	$(1,5,17)$
190	0.184	$(1,5,18)$	210	0.184	$(1,5,19)$	231	0.186	$(1,6,20)$
253	0.187	$(1,6,21)$	276	0.188	$(1,6,22)$	300	0.190	$(1,7,23)$
325	0.190	$(1,7,24)$	351	0.190	$(1,7,25)$	378	0.190	$(1,8,26)$
406	0.192	$(1,8,27)$	435	0.192	$(1,8,28)$	465	0.192	$(1,9,29)$
496	0.193	$(1,9,30)$	528	0.194	$(1,9,31)$	561	0.194	$(1,9,32)$
595	0.194	$(1,10,33)$	630	0.195	$(1,10,34)$	666	0.195	$(1,10,35)$
703	0.195	$(1,10,36)$	741	0.196	$(1,11,37)$	780	0.196	$(1,11,38)$
820	0.196	$(1,11,39)$	861	0.196	$(1,12,40)$	903	0.197	$(1,12,41)$
946	0.197	$(1,12,42)$	990	0.197	$(1,12,43)$			
15	0.125	$(2,1,2)$	35	0.179	$(2,1,3)$	70	0.179	$(2,1,4)$
126	0.214	$(2,2,5)$	210	0.222	$(2,2,6)$	330	0.229	$(2,3,7)$
495	0.240	$(2,3,8)$	715	0.241	$(2,3,9)$	1001	0.248	$(2,4,10)$
28	0.167	$(3,1,2)$	84	0.204	$(3,1,3)$	210	0.222	$(3,2,4)$
462	0.247	$(3,2,5)$	924	0.247	$(3,2,6)$			
45	0.188	$(4,1,2)$	165	0.216	$(4,1,3)$	495	0.244	$(4,2,4)$
66	0.200	$(5,1,2)$	286	0.223	$(5,1,3)$	1001	0.258	$(5,2,4)$
91	0.208	$(6,1,2)$	455	0.228	$(6,1,3)$			
120	0.214	$(7,1,2)$	680	0.231	$(7,1,3)$			
153	0.219	$(8,1,2)$	969	0.234	$(8,1,3)$			
190	0.222	$(9,1,2)$	231	0.225	$(10,1,2)$	276	0.227	$(11,1,2)$
325	0.229	$(12,1,2)$	378	0.231	$(13,1,2)$	435	0.232	$(14,1,2)$
496	0.233	$(15,1,2)$	561	0.234	$(16,1,2)$	630	0.235	$(17,1,2)$
703	0.236	$(18,1,2)$	780	0.237	$(19,1,2)$	861	0.238	$(20,1,2)$
946	0.238	$(21,1,2)$						

Table 4. Achievable Bounds for $\gamma+\delta=\log(XY)/\log(N)$

RSA PARAMETERS		
96 bytes	$p =$	0x99c72d76722f217240dea9dead9339ed0f610e47166a6b995c5b4dd62532f0bf 3c7a1875431f3b98af2b5fc284d13956e2f58a819ba9e5be28b300ae99a43c08 e601cf6da250cd2902dde345b90632201cc2eebfe776151a53409d87aa1f27a5
96 bytes	$q =$	0xca60434275ea27e7eedba94fac9947986569b321a8784c1b1283f2b3c62746f9 8ca4fbe00609750e11c239ca222a52c540ababba6133504ec3610556e46ba6ca a52f63dcb5f081ae58adfeb6ef41fc744528af66bfdb79a026631af8568fbcbd
192 bytes	$N =$	0x7990fcf78f986a4bd285a5f8c4ac98d546c98d95fea23f3034bfc306cc22d9c3 423743ecbc346add601a94eff298f84f5476896c704fd364121c0d9d6873cebe 3124339bd3298a98ce4da3d42efe3a863e25979613e7e357c17e248e928bf01f 1abe69e6cd0c761cc54a21edc60466e5c49fa0abdf57fabc21ec41ca51f58d7b 44f08666b408a0508c6e114efdbbc6de3c2d65d350d18720044d5410d1afbbb6 9dc0cf57da519aa7aec644b9f792e369cf6501359305b059475696a80d4870d1
1 byte	$e =$	0x03
96 bytes	$d =$	0x510b534fb51046dd3703c3fb2dc865e384865e63ff16d4cacdd52caf32c1e682 2c24d7f32822f1e8eabc634aa1bb5034e2f9b0f2f58a8ced616809139af7df29 76182267e21bb1bb3433c28d74a97c597ec3ba640d45423a80fec309b707f569 2464a61e98a21dd70e5fdf2a47e543958a8dea2cc04e2cafccb3562aef12392c 528ba160f1ea9fc687aafa818f2ad1d6bb081fba37f8360cbad0deb237bfe5ec b70a68090160328ae226ec7e34dc788e48fb975f47cd6bbf33cc941ab311084b
MESSAGE FORMATTING		
71 bytes	text =	Fault Attacks on RSA Signatures with Partially Unknown Message Encoding
81 bytes	$\alpha =$	O for a Muse of fire, that would ascend the brightest heaven of invention, a king
8 bytes	$r =$	0xb43558c5e4aeb6e8
81 bytes	$\alpha' =$	dom for a stage, princes to act and monarchs to behold the swelling scene! Then s
20 bytes	$H(\alpha\|\|r\|\|\alpha') =$	0xb27dd515171666e807e48cc7d811d91eb824cb27
192 bytes	$\mu(m) =$	0x6a4f20666f722061204d757365206f6620666972652c207468617420776f756c 6420617363656e6420746865206272696768746573742068656176656e206f66 20696e76656e74696f6e2c2061206b696e67b43558c5e4aeb6e8646f6d20666f 722061207374616765 2c207072696e63657320746f2061637420616e64206d6f 6e617263687320746f206265686f6c6420746865207377656c6c696e67207363 656e6521205468656e2073b27dd515171666e807e48cc7d811d91eb824cb27bc
FAULTY SIGNATURE (FAULT IN σ_p)		
96 bytes	$\sigma'_p =$	0x23ebbf4274295ed52ec83f6d67da3f08ead22455b956d6e8f4eadf3662462761 6c632b3899a64e0e2e5bec2d66f66e67146974fb441365dcab1a7a33e4fe4868 a9e8955fa2f2f0572c66333e6898e3e5ad9ed88665126d182ab3677b172733af
96 bytes	$\sigma_q =$	0x2e0505263d4e7eebbfd294f06ffb9cb00816730d30cc27a52918920d9bdcb9d8 08fc4cbddb5fae85dd40f50fef20aafbe73917e8daae4d7c9f96ac102bf64b80 e4947f3509ed11a659c62a6e5f24ed611e7dffc9e77fbd1a42f41e946be7561e
192 bytes	$\sigma' =$	0x5defad0dc0082ec43c48860dbc3cf4160a66deb6c2304cd1323920941ad5a184 bf92da34608e407042e8db9f3778b181f58f3fd3b6851ca0fd8fb975419c9f28 8d13f2f82a161be25a8fb27810f83a118f6a4f2f7e5cfc54582d88862ba3eec9 a3ceb7330fe4dde64c3f99ef23a07a78180dd43d2c455e52c3bc55553d25e36c f8265eb4bf4c7b9198a8b89d622ffcd52ed1702988f40602064bae6a7d599884 b8d23f5394d2b4aeecbd33d177f46baf71cdff70cafabfd0ac1ea2548c5ab5a6
ADDITIONAL DATA: CORRECT SIGNATURE (σ_q SAME AS ABOVE)		
96 bytes	$\sigma_p =$	0x8c0bd7487bd14111703e1c7d3d94cdbf6cc37f1c7f958d764e93285c3ef94c1e 8dae46a4fe1615b194515c3f646bd01a358756dad5abdab19bf95b118ce98002 fda41ae0cc583442cf7b50cea8740d1428c1451bb6d9daceb55d33dfb3363194
192 bytes	$\sigma =$	0x5a67520fb1a049a1b027905867280e66fadbdb2d375c05ea6678e77db836f991 48134a673e009d4e09e02e68513228d986f580340f3b474012d28fb0f0aa221c d5a6485d5a17ec0993240b923c4167ebd959bac0c3194bade0de8baa3e3782da 4981465e8d721709be01d70242cafdd6c7d031bb6ca32d9dececf4ad06fdeca8 5e5b6b6f63a4284241ffe5c9f8937499607ae67c3c5852663f39c48f8e69526d 475af6214f36d09e66b17268e2c3b674c08248592164f45328721e3f75c9c0ac

Table 5. Faulty 1536-bit RSA signature example (H = SHA-1).