Fault Attacks on Public Key Elements: Application to DLP-based Schemes

Chong-Hee Kim*, Philippe Bulens**,
Christophe Petit*** and Jean-Jacques Quisquater

UCL Crypto Group, Université Catholique de Louvain,
3 Place du Levant, 1348 Louvain-la-Neuve, Belgium,
TEL: +32 (0) 10 47 81 41, FAX: +32 (0) 10 47 25 98,
first.lastname@uclouvain.be

Abstract. Many cryptosystems suffer from fault attacks when implemented in physical devices such as smart cards. Fault attacks on secret key elements have successfully targeted many protocols relying on the Elliptic Curve Discrete Logarithm Problem (ECDLP), the Integer Factorization Problem (IFP) or the Discrete Logarithm Problem (DLP). More recently, faults attacks have also been designed against the *public* key elements of ECDLP and IFP-based schemes.
In this paper, we present the first fault attacks on the public key elements of DSA and ElGamal, two DLP-based signature schemes. Our attacks fully recover a 160-bit DSA secret key and a 1024-bit ElGamal secret key with $\sim 4 \cdot 10^7$ and $\sim 3 \cdot 10^6$ faulty signatures respectively. Such figures might suggest that DLP-based schemes are less prone to fault attacks than ECDLP- and IFP-based schemes. However, the integrity of public keys should always be checked in order to thwart such attacks since improvements may reduce the required amount of faulty signatures in the near future.

Index terms - Smart cards, side channel, fault injection, faults attacks, ElGamal, DSA

1 Introduction

From a classical point of view, cryptanalysis is an abstract mathematical notion. However in practice, algorithms have to be implemented on real physical devices that are exposed to side-channel attacks like timing attacks [11, 13, 26, 27, 29], power attacks [10, 19, 21, 25], electromagnetic attacks [1, 15, 24] as well as fault attacks [3, 8, 16].

After the discovery of fault attacks (1970's) [20], various mechanisms for fault production and propagation have been discovered and researched. Some of the most popular fault injection techniques include variations in supply voltage,

* Supported by Walloon Region, Belgium / E.USER project.
** Supported by Walloon Region, Belgium / First Europe Program.
*** Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

clock frequency, temperature or the use of white light, X-ray or ion beams [3].

It is not sufficient to just induce a fault in the cryptographic device during a calculation involving a secret. Depending on the algorithm, it is also crucial to generate the *right* fault at the *right* time and on the *right* place within the chip. The generated fault must be exploitable, meaning that the attacker has to be able to extract some information about the secret from the erroneous result of the algorithm. The first attack [8] that used a fault to derive secret information from a cryptosystem targeted a Chinese Remainder Theorem (CRT) based RSA implementation. After that, DES [5], RSA and ElGamal [2], LUC and Demytko [18], ECC [6], AES [7], and DSA [23] have also been compromised by fault attacks.

All these attacks targeted secret keys and computations involving secret keys. Recently, several papers have considered fault attacks on public key elements, demonstrating the necessity to protect public keys against fault attacks at least for ECDLP (Elliptic Curve Discrete Logarithm Problem) [4] and IFP (Integer Factorization Problem) based algorithms [9, 22, 28]. Curiously and to the best of our knowledge, no fault attack on public key elements has been proposed so far for any DLP (Discrete Logarithm Problem)-based schemes.

In this paper, we present the first fault attacks on the public key elements of two DLP-based algorithms, namely the ElGamal and DSA signature schemes. We also estimate the complexity of our attacks (the number of faults required) both by a theoretical analysis and software simulations. Although it turns out that our attacks require a large number of faults, they highlight the necessity to check the integrity of both public and private keys elements for DLP-based schemes.

Fault attack model: In our model, an attacker tries to corrupt not the private but **the public keys** by faults. Whereas some attacks [2, 23] require that only a single bit is flipped or a particular byte is set to zero, our model only assumes that the attacker is able to enforce random register faults. While such single or particular bit flips are rather hard to achieve in practice, changing a word or many words in an undetermined way is the most simple fault to induce. It can simply be obtained by inducing a fault on address decoders for example, when parameters stored in EEPROM or Flash are transferred to RAM. This results in a random transient change in the public key.

Outline of the paper: The rest of this paper is organized as follows. Section 2 describes previous fault attacks on ECDLP and IFP schemes with a corruption of public keys. In Sections 3 and 4 we present our new attacks on DSA and ElGamal signature scheme and describe their complexity by a theoretical analysis and software simulations. Section 5 contains some open discussions to improve the attacks, and Section 6 concludes the paper.

2 Related Works

Previous attacks on public key elements of cryptographic schemes were performed on ECDLP [4, 12] or IFP [28, 22, 9] cryptosystems.

2.1 Previous Attacks on ECDLP

Here, the attacker tries to shift the computation from a given secure elliptic curve E (in Weierstrass parameterization) to an insecure curve E^*. Consider a smart card that has to compute dP, for d a scalar and P a point on the curve

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

If a fault is induced on P, it is changed into a point P^* on a curve

$$E^* : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + {a_6}^*$$

The attack exploits the fact that the parameter a_6 is not used in the usual point addition formula on Weierstrass elliptic curves. As a consequence, the whole computation is performed on the curve E^*, and dP^* is obtained. If now E^* is an insecure curve (typically because ord(E^*) has a small factor), the discrete logarithm problem can be solved on it, which gives information about d. Faults occurring during the computation of dP are also exploitable [4]. The attack here assumes that only a few error bits are inserted in order to be successful. This hypothesis has been relaxed in [12] where a random and unknown fault either in the base point P, in the base field F_p or F_{2^q} underlying the curve, or in the curve's parameters is turned into breaking the scheme.

2.2 Previous Attacks on IFP

Seifert's RSA attack [28]: The attacker tries to corrupt RSA public key N into N^* during RSA signature verification by faults, where N^* is prime. Then he can simply compute the private exponent d^*, as e^{-1} mod $(N^* - 1)$, assuming that e is relatively prime to $(N^* - 1)$. In the off-line part, he can choose bits of N to modify the creation of N^*. In the on-line part, he repeatedly queries the device with a specially constructed message-signature pair and causes data faults until this particular N^* is used as the modulus in the signature verification algorithm.

Seifert proved that there exists an algorithm that will be successful with probability at least $\mathcal{O}(1/k)$ in getting fraudulent programs S^* authenticated and therefore executed on a machine relying on RSA-authentication of programs. The running time of the algorithm is $log^{\mathcal{O}(1)}(N)$, where N and e are an k-bit RSA public keys.

Generalization of Seifert's RSA attack [22]: Muir generalized Seifert's attack to moduli N^* not necessarily primes but still easy to factorize.

He simplified the analysis of Seifert's attack. He also showed an experimental result for the off-line stage of RSA-Attack. His analysis and computational trials show that if an adversary is able to cause random faults in *only 4 bits* of a 1024-bit RSA modulus stored in a device, then there is more than 50% chance that the modified modulus N^* has good property to attack. For Seifert's original attack, it required 8 bits.

Brier et al.'s RSA attack [9]: The attacker tries to corrupt RSA public key N into N^* similar to Seifert's attack, but here he tries to attack signature generation process, $s = m^d \bmod N$. Moreover, at the end of the attack the attacker has the secret key d.

The basic idea is based on the fact that, for small moduli - for example, from 15 to 20 digits -, discrete logarithms are efficiently computed by square root methods such as *baby-step giant-step* or *Pollard's rho*. In particular if $p^a | N^*$ and r is a small number dividing the multiplicative order of m modulo p^a, then from $s^* = m^d \bmod N^*$, $d \bmod r$ can be recovered. Gathering some fault couples (m_i, s_i) corresponding to unknown moduli ${N_i}^* \neq N$, the attacker retrieves the private exponent d off-line by progressively determining $d \bmod r_j$ for some small prime powers r_j. Once the product $R = \prod_k r_j$ exceeds the modulus N (and so unknown $\varphi(N)$), d is recovered with the Chinese Remainder Theorem.

Even when the adversary does not have any information on ${N_i}^*$, he can recover a 1024-bit secret d with about 60000 faults, and he needs only 28 faults once he knows the faulty ${N_i}^*$ values.

3 A New Fault Attack on DSA

3.1 The Digital Signature Algorithm

The system parameters for DSA [17] are $\{p, q, h, g\}$, where p is prime (at least 512 bits), q is a 160-bit prime dividing $(p-1)$, h is a hash function and $g \in \mathbb{Z}_p^*$ has order q. The private key is an integer $x \in \mathbb{Z}_q^*$ and the public key is $y = g^x \bmod p$.

Signature: To sign a message m, the signer picks a random $k < q$ and computes:

$$u \leftarrow (g^k \bmod p) \bmod q \qquad \text{and} \qquad v \leftarrow \frac{h(m) + xu}{k} \bmod q.$$

The signature of m is the pair (u, v).

Verification: To check (u, v) the verifier ascertains that:

$$u = (g^{wh(m)} y^{wu} \bmod p) \bmod q, \qquad \text{where } w = v^{-1} \bmod q.$$

3.2 A New Attack by a Fault Induction on p and q

Let us suppose that an attacker succeeded in invoking transient faults before starting signature generation and changing the values of p, q into p^*, q^*, respectively. Assuming the generated k satisfies $\gcd(k, q^*) = 1$, he gets:

$$u^* \leftarrow \left(g^k \bmod p^*\right) \bmod q^* \qquad \text{and} \qquad v^* \leftarrow \frac{h(m) + xu^*}{k} \bmod q^*.$$

Consider any integer t dividing p^* and q^*, such that $\varphi(t)$ divides q^* and g is not a root of zero modulo t. Then, we have

$$(u^*)^{v^*} \equiv g^{h(m)+xu^*} \bmod t. \tag{1}$$

Consequently,

$$\frac{(u^*)^{v^*}}{g^{h(m)}} \equiv (g^{u^*})^x \bmod t. \tag{2}$$

Denote $DL(\alpha, \beta, t)$ the discrete logarithm of $\beta \bmod t$ with respect to α. For any r_j dividing the multiplicative order of (g^{u^*}) modulo t we have

$$x \equiv DL(\alpha, \beta, t) \bmod r_j, \tag{3}$$

where $\beta = \frac{(u^*)^{v^*}}{g^{h(m)}} \bmod t$ and $\alpha = g^{u^*} \bmod t$ can be computed from the faulty signature.

Assuming the attacker knows the value of the faulty (p^*, q^*) induced, this suggests that an algorithm can recover the secret key x, by successively recovering x modulo r_j with various faulty signatures until the lowest common multiple of those r_j is large enough to recover x using the Chinese Remainder Theorem.

Toy example Take $p = 124540019$, $q = 17389$, $g = 10083255$. Let the private key be $x = 12496$, so $y = g^x \bmod p$. The message to be signed verifies $h(m) = 5246$, and $k = 9557$ is chosen. The signature is the pair $(u = 34, v = 13049)$. Now suppose that the attacker succeeded in making faults and changing the value of p into p^* and q into q^* before generating the signature. Then, we find x with the values shown in Table 1 .

Table 1. Toy example of the attack on DSA

p^*	q^*	k	u^*	v^*	t	α	β	r_j	$x \bmod r_j$	$\mathrm{lcm}(\{r_j\})$
124539997	17030	41	13384	9140	131	33	28	65	16	65
124539973	17110	6171	12501	422	59	47	7	58	26	3770
124539983	16536	9961	11694	1214	53	20	36	26	16	3770
124539989	17296	491	6434	202	47	24	18	23	7	86710

3.3 Analysis of the Attack

We now provide an estimation of the number of faults needed to recover the whole secret x. Our analysis will be completed in four steps. We first estimate the probability P_t that for a given t, the condition

$$C_t := (t \mid q^*) \wedge (\varphi(t) \mid q^*) \wedge (t \mid p^*) \wedge (\gcd(k, q^*) = 1) \tag{4}$$

is satisfied. Then, we approximate by 1 the probability $P_{r|t}$ that when this condition is satisfied for t, the secret x is recovered modulo r, where r is a prime power dividing $\phi(t)$. From these two probabilities we derive a global estimate for the probabilities $P[r_j]$ to recover x modulo r_j for various r_j, and we finally estimate the number of faults required from these probabilities. Each step in the analysis induces an error by some small factor in the estimations, so our final estimations only give an upper bound on the number of faults required, precise up to a small factor.

We first estimate $P_t = \Pr[C_t]$ where the probability is on random independent choices of k, p^* and q^*. Let write l_t for $\mathrm{lcm}(t, \varphi(t))$ and decompose q^* as $q^* = q_1^* l_t + q_2^*$. Then

$$P_t = \Pr[q_2^* = 0] \Pr[t \mid p^*] \Pr[(\gcd(k, q_1^*) = 1) \wedge (\gcd(k, l_t) = 1)].$$

Clearly, $\Pr[q_2^* = 0] = \frac{1}{l_t}$ and $\Pr[t \mid p^*] = \frac{1}{t}$. Neglecting correlations between the conditions on the gcds, we have

$$\Pr[(\gcd(k, q_1^*) = 1) \wedge (\gcd(k, l_t) = 1)] \approx \Pr[\gcd(k, q_1^*) = 1] \Pr[\gcd(k, l_t) = 1].$$

As $\Pr[\gcd(k, l_t) = 1] = \frac{\varphi(l_t)}{l_t}$ and $\Pr[\gcd(k, q_1^*) = 1] \approx \lim_{Q \to \infty} \sum_{q=0}^{Q} \frac{\varphi(q)}{q} = \frac{6}{\pi^2} \approx 0.6$ we finally get

$$P_t \approx 0.6 \frac{\varphi(l_t)}{t l_t^2} = 0.6 \frac{\varphi(\mathrm{lcm}(t, \varphi(t)))}{t(\mathrm{lcm}(t, \varphi(t))^2}. \tag{5}$$

The conditions on the gcds are actually dependent with a positive correlation: clearly if $\gcd(k, q_1^*) = 1$ then for example k is more likely to be a prime and then $\gcd(k, l_t) = 1$ will be satisfied also. However, as

$$\frac{\Pr[(\gcd(k, q_1^*) = 1) \wedge (\gcd(k, l_t) = 1)]}{\Pr[\gcd(k, q_1^*) = 1] \Pr[\gcd(k, l_t) = 1]} \leq \frac{\Pr[\gcd(k, q_1^*) = 1]}{\Pr[\gcd(k, q_1^*) = 1] \Pr[\gcd(k, l_t) = 1]} = \frac{l_t}{\varphi(l_t)},$$

for most t values the actual probability is only a small factor larger than our approximation.

Now suppose a fault is performed such that Condition (4) is satisfied. For each r dividing $\varphi(t)$, we evaluate the probability $P_{r|t}$ that the attacker recovers $x \bmod r$ from Equation (2), *i.e.* by computing a discrete logarithm modulo t.

As p^* is uniformly random, the remainder of $u^* = g^k \bmod p^*$ modulo $\varphi(t)$ is nearly uniformly distributed in $\mathbb{Z}_{\varphi(t)}$. We deduce that α is nearly uniformly

distributed in the subgroup of $\mathbb{Z}_t^*$ generated by g. Write $r_{g,t} = \prod_{i=1}^{I} p_i^{e_i}$ and $r_{\alpha,t}$ for the multiplicative orders of g and α modulo t, where p_i are distinct primes. Then, for any divisor $r = \prod_{i=1}^{I} p_i^{e'_i}$ of $r_{g,t}$, the probability that $r_{\alpha,t} = r$ is $\frac{\varphi(r)}{r_{g,t}} = \prod p_i^{-e_i} \prod_{e'_i \neq 0} (p_i - 1) p_i^{e'_i - 1}$, and the probability that r divides $r_{\alpha,t}$ is $P_{r|t} = \prod_{e'_i \neq 0} p_i^{-(e_i - e'_i + 1)} (p_i^{e_i - e'_i + 1} - 1)$.

In particular if r is a power of a prime, *i.e.* $r = p_r^{e'_r}$, then $\frac{1}{2} \leq P_{r|t} = p_r^{-(e_r - e'_r + 1)} (p_r^{e_r - e'_r + 1} - 1) \leq 1$. We approximate this probability by 1 for all r, which is a good approximation for all but very small r. The approximation means the following: we suppose that once a fault that satisfies Condition (4) for some t is induced, the attacker recovers x modulo r for any r dividing $\varphi(t)$.

For any set of integers T, write P_T for the probability that all $t \in T$ verify Condition (4), and no other integer. We now evaluate $P[r]$, the probability to recover the secret modulo a prime power r if the attack is performed once. According to our previous arguments, we have

$$P[r] \approx \sum_{\substack{T = \{t_i\} \\ \text{s.t. } \exists t \in T \text{ s.t. } r | \varphi(t)}} P_T.$$

We finally approximate

$$P[r] \approx \sum_{\substack{T = \{t_i\} \\ \text{s.t. } t_r \in T}} P_T = P_{t_r} \qquad \text{where } t_r = \arg \max_{t \text{ s.t. } r | \varphi(t)} P_t. \tag{6}$$

This approximation underestimates the actual probability because all the sets in the original sum that do not contain t_r are neglected. However, according to the estimation (5), P_t tend to decrease very fast with t, so for the sets T that do not contain t_r the probability P_T is much smaller than P_{t_r}, and the actual probability $P[r]$ is only a small factor larger than our estimation.

We now estimate the number of bits of the secret key that can be recovered for a given amount of faults. In order to recover x by means of the Chinese Remainders Theorem, an adversary should know a list of values $x \bmod r_j$ such that $\mathrm{lcm}(\{r_j\}) > x$. On the other hand, if N faults are performed, it is likely that x will be recovered modulo r_j for all r_j such that $P[r_j] > N^{-1}$, so the quantity of information that can be recovered with N faults is

$$\log_2(\mathrm{lcm}(\{r_j | P[r_j] \geq N^{-1}\})). \tag{7}$$

This conclude our analysis.

In Figure 1, we display the probabilities $P[r_j]$ evaluated from Equations (5) and (6) for every r_j power of prime up to 5000. Using this computation, we further estimate for any $N_s = 10^s$ with $4 \leq s \leq 10$, the number of bits of the

secret key that can be recovered. The results are presented in Figure 2. The last row in the table corresponds to 160-bits security and predicts that about $2.14 \cdot 10^8$ million faults are needed to recover the secret. The table and the figure additionally show that, at least for this range of parameters, the attack complexity is between cubic and biquadratic in the size of the secret.

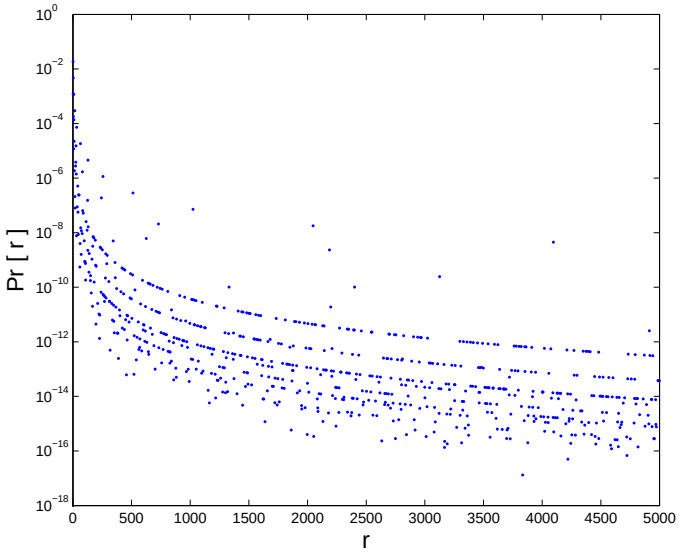

Fig. 1. Approximation (6) for all r power of prime smaller than 5000.

3.4 Experimental Results

In order to check the soundness of our theoretical analysis, we simulated the attack in software (with parameters p a 1024-bit prime, q a 160-bit prime and g about the size of p). The results are shown in Table 2, where each column gives the amount of faulty couples (p^*, q^*) used to retrieve m bits of the private key x. As can be seen, the required amount of faults in our simulations is 3 to 10 times smaller than the $2.14 \cdot 10^8$ theoretical bound with a mean of $4.16 \cdot 10^7$ faults.

$N_s = 10^s$	$\lvert x\rvert \approx \log_2(\mathrm{lcm}(\{r_j\}))$
10^4	9
10^5	19
10^6	38
10^7	61
10^8	122
10^9	253
10^{10}	472
$2.14 \cdot 10^8$	160

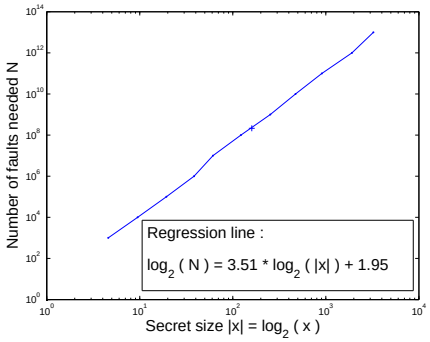

Fig. 2. Following the theoretical analysis of our attack on DSA, the number of faults needed N is between cubic and biquadratic with respect to the secret size $|x|$.

Table 2. Seven experimental software results for our attack on DSA. In each column, we display the number of faults needed to recover m bits of the secret for each of the experiments.

Exp. n°	$m = 32$	$m = 64$	$m = 96$	$m = 128$	$m = 160$
1	463986	5676441	8549083	26476140	38621903
2	811215	8898520	14945495	22174790	34861119
3	284706	3336328	5577267	11579152	20960118
4	798230	6438425	21496166	31049856	61711260
5	282811	8363054	14303797	26594960	39066576
6	721742	8762969	20601681	38398403	73933924
7	453623	3174393	10220088	17145832	22623221

4 A New Fault Attack on ElGamal Signature Algorithm

4.1 ElGamal Signature Algorithm

In the ElGamal signature scheme [14], to generate a private and public key pair, we first choose a prime p, and two random numbers, g and x, such that both g and x are less than p. The private key is x and the public key is $(y = g^x \bmod p, g, p)$.

Signature To generate a signature on a message m, the signer first picks a random k such that k is relatively prime to $(p-1)$. He then computes

$$u \equiv g^k \bmod p \qquad \text{and} \qquad v \equiv \frac{m - xu}{k} \bmod (p-1).$$

Verification The signature is the pair (u, v) and is verified by checking that

$$y^u u^v \equiv g^m \bmod p.$$

4.2 A New Attack by Fault Injection on p^*

Suppose that there are faults before the signature generation and the modulus p is changed into p^*. Then, if $\gcd(k, p^* - 1) = 1$, we have

$$u^* \equiv g^k \bmod p^* \qquad \text{and} \qquad v^* \equiv \frac{m - xu^*}{k} \bmod (p^* - 1).$$

Let t divide p^* and $\varphi(t)$ divide $(p^* - 1)$. Then, we have

$$(u^*)^{v^*} \equiv g^{(m - xu^*)} \bmod t. \tag{8}$$

Consequently, if g is not a root of 0 modulo t,

$$\frac{(u^*)^{v^*}}{g^m} \equiv (g^{-u^*})^x \bmod t. \tag{9}$$

For any r_j dividing the multiplicative order of $(g^{-u^*}) \bmod t$, we have

$$x \equiv \mathrm{DL}(\alpha, \beta, t) \bmod r_j, \tag{10}$$

where $\beta = \frac{(u^*)^{v^*}}{g^m} \bmod t$ and $\alpha = g^{-u^*} \bmod t$ can be computed from the faulty signature.

Assuming that the attacker knows the value of the faulty p^* induced, equation (10) makes it possible to recover some information about x. By retrieving $x \bmod r_j$ for sufficiently many r_j satisfying $\mathrm{lcm}(\{r_j\}) > x$, the whole secret can be recovered by use of Chinese Remainder Theorem.

4.3 Analysis and Experimental Results

The analysis of ElGamal case is similar to (and actually much easier than) DSA case, so we omit its detail. In the table of Figure 3, we computed $\log_2(\mathrm{lcm}(\{r_j\}))$ for all r_j that have probability larger than $b_s = 10^{-s}$ for $4 \leq s \leq 10$. The last row in the table corresponds to 1024-bits security and predict that about three million faults are needed to recover the secret. The table and the figure additionally show that, at least for this range of parameters, the attack complexity is a bit more than quadratic in the size of the secret.

$N_f \approx b_s^{-1}$	$\lvert x\rvert \approx \log_2(\mathrm{lcm}(\{r_j\}))$
10^4	63
10^5	218
10^6	641
10^7	1788
10^8	5319
10^9	16019
10^{10}	47468
$2.89 \cdot 10^6$	1024

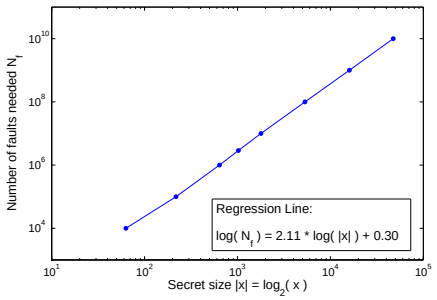

Fig. 3. Following the theoretical analysis of our attack on ElGamal, the number of faults needed N_f is a bit more than quadratic with respect to the secret size $|x|$.

Table 3. Six experimental software results with increasing factorization bound for our attack on ElGamal. In each column, we display the number of faults needed to recover m bits of the secret for each of the experiments.

Exp n°	Bound on t	$m = 64$	$m = 128$	$m = 256$	$m = 512$	$m = 1024$
1	10^4	5794	39790	104258	786038	3366136
2	$2 \cdot 10^4$	4902	24054	126230	611494	2893428
3	$5 \cdot 10^4$	4902	24054	126230	571170	2688882
4	$5 \cdot 10^4$	8810	21742	102974	572170	2931512
5	$5 \cdot 10^4$	7420	23290	142810	696046	3011938
6	$10 \cdot 10^4$	3078	43274	135194	636864	2808468

Table 3 exhibits the amount of faulty primes generated in order to recover m bits of the private key x in various simulations. In these experiments, each faulty p^* was factorized up to a bound that is indicated in the first column of the table (our predictions considered a bound of $5 \cdot 10^4$). As can be seen, the practical results match pretty well the theoretical expectations. Moreover, the table shows that considering more factors of p^* slightly decreases the numbers of faulty primes required to find the key.

5 Open Discussion and Future Directions

We have shown both theoretically and by software simulation that fault attacks on public key elements are successful against the DLP-based DSA and ElGamal signature schemes. Our attacks are however not yet practical because they require that the attacker guesses the faulty values he induced, and because the quantities of faulty signatures needed are too large for practical applications.

5.1 Finding the Faulty Value on the Public Key Elements

Blind Method [9] As we have seen above, if Condition (4) is verified for (p^*, q^*), we recover x modulo r from the discrete logarithm of β in basis α. However, if the adversary does not know the faulty values, he cannot check whether Condition (4) holds. What he can do, is storing all computed values and distinguish the correct one by a statistical method. We point out that the discrete logarithm of β with respect to α may not exist. In this case, the attacker knows that Condition (4) is not verified and simply discards the fault. When the condition does not hold but the logarithm exists, we assume that all values are as likely for $x \bmod r$. Consequently, a bias-based attack may be built for DSA. We remark however, that due to the small probability for Condition (4) to occur, the bias will be much smaller. The same remark holds for our attack against ElGamal.

Collision Method [9] This method requires a different model of faults, namely a *Dictionary* of possible faulty values. For our attack on DSA, this dictionary $\mathcal{D}$ would of course be two-dimensional. In [9], the authors use the notion of *marker* of an element of the dictionary, which would be defined as couples (t, r) for DSA, where $(t \mid q^*) \wedge (\varphi(t) \mid q^*) \wedge (t \mid p^*)$ and r is a not too small factor of $\varphi(t)$. For every fault $(p^*, q^*) \in \mathcal{D}$ and every marker (t, r), the attacker keeps the discrete logarithm $DL(\alpha, \beta, t, r)$. As soon as two identical values are found, this common value is believed to be the true value, that is $x \bmod r$. In order to avoid false positives, $\sqrt{r}$ should be much larger than $\mid \mathcal{D} \mid$ [9], so t must be much larger than $\mid \mathcal{D} \mid^2$.

We argue that this method cannot be applied in the context of our attack on DSA, simply because most likely, no element of the dictionary will have an appropriate marker. Indeed, the probability that a random element satisfies $(t \mid q^*) \wedge (\varphi(t) \mid q^*) \wedge (t \mid p^*)$ is $\frac{1}{t \cdot \operatorname{lcm}(t, \varphi(t))}$ so the probability that an element of the dictionary has a marker (t, r) for some $t > \mid \mathcal{D} \mid$ is smaller than

$$\sum_{t=|\mathcal{D}|^2}^{\infty} \frac{1}{t \cdot \operatorname{lcm}(t, \varphi(t))} \leq \sum_{t=|\mathcal{D}|^2}^{\infty} \frac{1}{t^2} \approx \frac{1}{\mid \mathcal{D} \mid^2}.$$

The same conclusion holds for ElGamal.

Optimal method [9] This method uses the whole information available from faults, in the sense that it keeps all possible values for the sequence of faults $((p_1^*, q_1^*), (p_2^*, q_2^*), (p_3^*, q_3^*), \ldots)$ that are coherent with the faulty signatures $((u_1, v_1), (u_2, v_2), (u_3, v_3), \ldots)$. Coherency conditions include for example that $\gcd(g, p^*, q^*)$ divides u^*. Moreover it is "optimal" in the sense that it associates a selectivity parameter related to a "maximum-likelihood selection" to each possible sequence of faults.

In theory, this method can be applied to our attacks on DSA and ElGamal. However, handling the lists of fault values will most likely become intractable in practice.

Further Methods By relaxing the condition $\sqrt{r} \gg \mid \mathcal{D} \mid$ we can allow some false positives in the collision method (and discriminate between false and true positives via coherency conditions). In our attack on DSA, we could also think of having a dictionary for p^* only. Furthermore, we could use a fault model in which an induced error occurs only on a few bytes instead of on the whole value. This is reasonable because a byte or a word is a basic unit for communication and computation in modern embedded devices such as smart cards. Therefore we can reduce the possible candidates for faulty values.

5.2 Reducing the Required Number of Faulty Signatures

As $\mathrm{lcm}(\{r_i\})$ should be larger than the size of the secret key, the required number of faulty signatures dramatically increases with the secret key size. To partially solve this problem, we can think of a hybrid method recovering part of the secret key with our attack and the remaining bits with an exhaustive search. In this case, we should find the optimal number of bits by considering the trade-off between the required number of faulty signatures and the amount of computation for the exhaustive search.

6 Conclusion

For the first time in the literature, we have shown that fault attacks on two DLP-based schemes, DSA and ElGamal signature schemes, are possible by inducing faults on the public key. Both attacks were carefully analyzed in order to derive their probabilities of success. In each case, the analysis was validated by software simulation. The amount of faulty signatures needed to recover a 160-bit DSA secret key is about $\sim 4 \cdot 10^7$, and $\sim 3 \cdot 10^6$ for a 1024-bit ElGamal.

In order to render the attack more practical, some improvements are required to reduce the number of faulty signatures and to ease the finding of the faults induced. But fault attacks against DLP-based schemes are meant to improve over time, we therefore recommend checking the integrity of public parameters whatever the underlying number theoretic problem, that is in the case of DLP-based scheme as much as in the case of ECDLP- or IFP-based schemes.

References

1. D. Agrawal, B. Archambeault, J. Rao, and P. Rohatgi. The EM side-channel(s). In *Cryptographic Hardware and Embedded Systems – CHES'02*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.
2. F. Bao, R. Deng, Y. Han, A. Jeng, A. Narasimhalu, and T. Ngair. Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. In *Proceedings of the 5th Workshop on Secure Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 115–124. Springer, 1997.
3. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. In *Fault Diagnosis and Tolerance in Cryptography in association with DSN 2004 – The International Conference on Dependable Systems and Networks*, pages 330–342, 2004.
4. I. Biehl, B. Meyer, and V. Muller. Differential fault attacks on elliptic curve cryptosystems. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2000.
5. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Proceedings of the 17th annual international cryptology conference on advances in cryptology*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.
6. J. Blömer, M. Otto, and J.-P. Seifert. Sign change fault attacks on elliptic curve cryptosystems. In *Fault Diagnosis and Tolerance in Cryptography – FDTC'05*, pages 25–40, 2005.
7. J. Blömer and J.-P. Seifert. Fault based cryptanalysis of the advanced encryption standard (AES). In *Financial Cryptography – FC'03*, volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer, 2003.
8. D. Boneh, R. DeMillo, and R. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14(2):101–119, 2001. An earlier version appears in [**?**].
9. E. Brier, B. Chevallier-Mames, M. Ciet, and C. Clavier. Why one should also secure RSA public key elements. In *Cryptographic Hardware and Embedded Systems – CHES'06*, volume 4249 of *Lecture Notes in Computer Science*, pages 324–338. Springer, 2006.
10. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems – CHES'04*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
11. D. Brumley and D. Boneh. Remote timing attacks are practical. In *Proceedings of the 12th Usenix Security Symposium*, pages 1–14, 2003.
12. M. Ciet and M. Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *Design, Codes and Cryptography*, 36(1):33–43, 2005.
13. J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems. A practical implementation of the timing attack. In *Smart Card Research and Advanced Applications - CARDIS'98*, volume 1820 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 1998.
14. T. ElGamal. A public key cryptosystems and a signature scheme based on Discrete-Logarithm. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.
15. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems – CHES'01*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

16. C. Giraud and H. Thiebeauld. A survey on fault attacks. In *Smart Card Research and Advanced Applications VI - 18th IFIP World Computer Congress*, pages 159–176. Kluwer Academic Publishers, 2004.
17. N. institute of standards and technology. *Digital Signature Standard*. NIST FIPS PUB 186-2, 2000.
18. M. Joye and J.-J. Quisquater. A new and optimal chosen message attack on RSA-type cryptosystem. In *Information and Communication Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 302–313. Springer, 1997.
19. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology – CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
20. T. May and M. Woods. A new physical mechanism for soft errors in dynamic memories. In *Proceedings of the 16-th International Reliability Physics Symposium*, 1978.
21. T. Messerges, E. Dabbish, and R. Sloan. Examining smart-card security under the threat of power analysis attack. *IEEE Transactions on Computers*, 51(5):541–552, 2002.
22. J. Muir. Seifert's RSA fault attack: simplified analysis and generalizations. IACR Eprint archive 2005.
23. D. Naccache, P. Nguyen, M. Tunstall, and C. Whelan. Experimenting with faults, lattices and the DSA. In *Public Key Cryptography – PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 2005.
24. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and countermeasures for smart cards. In *International Conference on Research in Smart Cards, E-smart 2001*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
25. J.-J. Quisquater and D. Samyde. Automatic code recognition for smartcards using a kohonen neural network. In *Proceedings of the Fifth Smart Card Research and Advanced Application Conference (CARDIS '02)*, 2002.
26. W. Schindler. A timing attack against RSA with the chinese remainder theorem. In *Cryptographic Hardware and Embedded Systems – CHES'00*, volume 1965 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2000.
27. W. Schindler, J.-J. Quisquater, and F. Koeune. Improving divide and conquer attacks against cryptosystems by better error detection correction strategies. In *Proc. of 8th IMA International Conference on Cryptography and Coding*, pages 245–267, 2001.
28. J.-P. Seifert. On authenticated computing and RSA-based authentication. In *Proc. of ACM conference on computer and communications security 2005*, pages 122–127, 2005.
29. C. Walter and S. Thompson. Distinguishing exponent digits by observing modular subtractions. In *Cryptographer's Track at RSA conference – CT-RSA'01*, volume 2020 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2001.