Fault Attacks on ElGamal-Type Cryptosystems

Martin Kreuzer[1] [0000−0002−4732−2627], Nourhan Elhamawy[2], Maël Gay[2] [0000−0002−7640−7232], Ange-Salome Messeng Ekossono[1], and Ilia Polian[2] [0000−0002−6563−2725]

[1] Fakultät für Informatik und Mathematik, Universität Passau, D-94030 Passau, Germany {Martin.Kreuzer,Ange-Salome.MessengEkossono}@uni-passau.de
[2] Institut für Technische Informatik, Universität Stuttgart, Pfaffenwaldring 47, D-70569 Stuttgart, Germany
{Nourhan.Elhamawy,Mael.Gay,Ilia.Polian}@informatik.uni-stuttgart.de

Abstract. ElGamal-type cryptosystems are based on the difficulty of solving the Diffie-Hellman problem in suitable groups, e.g., in the group $(\mathbb{F}_p)^\times$ of units of a finite field or the group $E(\mathbb{F}_p)$ of $\mathbb{F}_p$-rational points on an elliptic curve. This paper studies fault-injection attacks against the private key of these cryptosystems. It extends previously introduced bit and byte fault models by several, increasingly less restrictive, fault models that assume faults in s-bit portions (subtuples) of the key. We provide a mathematical analysis and algorithms to compute the key subtuples satisfying the restraints of a number of fault injections, and use software simulations to determine the average number of necessary fault injections for all fault models. A hardware implementation of the fault attacks for the elliptic curve case shows that generating the raw data for these attacks is absolutely practical.

Keywords: Fault attack · fault injection · elliptic curve cryptography · ElGamal cryptosystem

1 Introduction

The ongoing transition to cyber-physical and autonomous systems has led to a strong need for high-performance and yet low-cost security solutions for such systems. As a consequence, hardware implementations of cryptographic primitives are becoming increasingly attractive due to their high speed and low energy requirements. A central requirement for cryptographic hardware is its resistance against physical attacks, both passive side-channel analysis [34, 37] and active fault-injection attacks [6, 40]. To design appropriate countermeasures and validate their effectiveness, it is necessary to accurately understand the threat and the potential of an adversarial party to mount an attack.

In this paper, we investigate fault-injection attacks on ElGamal-type cryptosystems. In 1985, Taher ElGamal suggested a cryptosystem based on the difficulty to compute discrete logarithms in the group $(\mathbb{Z}/p\mathbb{Z})^\times$, where p is a large prime number [19]. Later this cryptosystem was generalized to other groups with a difficult Discrete Logarithm Problem (DLP), for instance the groups $E(\mathbb{F}_p)$ of

$\mathbb{F}_p$-rational points on an elliptic curve [33], matrix groups over non-commutative rings [38], [29], or unit groups of group rings [26]. While many symmetric cryptosystems have been shown to be vulnerable to fault attacks requiring only a few fault injections (see e.g. [12]), asymmetric schemes tend to require larger numbers of injections or higher capabilities of the attacker. The main goal of this paper is to study the relations between these capabilities and the average number of required fault injections for ElGamal-type cryptosystems.

Since the encryption function of a public key cryptosystem does not use secret information, it is natural to attack the decryption function. Let us briefly recall the construction of the ElGamal-type cryptosystem based on a group G. For simplicity, we assume that G is abelian and use the additive notation. The publicly known information consists of G, and element $P \in G$ (typically of large prime order), and an element $Q \in \langle P \rangle$. The secret key is a positive integer a which was used to compute $Q = aP$. To encrypt a message $M \in G$, Bob chooses a random positive integer k, computes $C_1 = kP$ and $C_2 = M + kQ$, and sends (C_1, C_2) to Alice. To decrypt this message, Alice calculates $M = C_2 - aC_1$. Clearly, to mount a fault attack, we need to disturb this calculation.

Some of the first fault attacks on the classical ElGamal cryptosystem assumed that the attacker is able to flip a random bit of a during the calculation of aC_1 (see [4], [16]). We will reconsider this assumption as our fault model FM1.

Other fault attacks have targeted the specific implementation of the calculation of $C_2 - aC_1$. For instance, [36] attacked the classical case and [8] the case of the Elliptic Curve ElGamal (ECEG) cryptosystem by feeding invalid ciphertexts into the decryption algorithm such that the output values reveal information about the secret key. Specifically for the ECEG cryptosystem, fault models have been used which introduce transient errors into the scalar multiplication [8], [14], but they, too, produced points not on the original curve and were easy to avoid. Further attacks were devised in [9] and extended in [21] to attack particular algorithms for computing aC_1, namely by repeated doubling or the Montgomery ladder. However, they appear to be dependent on the specific implementation of the computation of aC_1 and apply mainly to the ECEG cryptosystem. Yet another class of fault attacks has targeted the public keys, for instance [11], [7] for the classical case and [31] for the elliptic curve case.

In this paper, we extend the analysis of fault attacks on ElGamal cryptosystems which assume fault injections into the secret key. Generalizing the bit flip fault model, we consider fault injections into a fixed s-bit block of the secret key via the fault model FM2. For byte faults, i.e., for the case $s = 8$, this fault model has already been studied carefully in [23] and [24]. By varying the value of s, we get an interesting trade-off: choosing a smaller s requires (linearly) more chunks to be attacked, but the required number of injections is exponentially smaller. This suggests that as small an s as can be reliably supported by the fault injection equipment should be used. Moreover, a careful analysis of the information revealed by a single fault injection allows us to provide optimal bounds for the numbers of necessary fault injections. In particular, we reconfirm the optimality of the bound for $s = 8$ given in [24].

A further weakening of the power of the attacker is assumed in the fault model FM3: random faults are injected into s-blocks of the secret key, but the temporal resolution does not suffice to control which s-block is hit. In this case, the number of necessary fault injections can be derived from FM2 in a straightforward way and is not much higher. However, the analysis of our fourth fault model FM4 is somewhat more intricate: we assume a "sliding window" of s-blocks of the secret key into which random faults are injected, but no control over which block is affected. Even in this case, we obtain realistic attack scenarios for practically relevant cases.

Thus the attacks discussed in this work are based on fault models with different assumptions on the adversary's capabilities to inject faults. One can generally distinguish between low-cost, low-precision injection techniques (like under-powering or inducing clock glitches [5]) and elaborate optical [43] and electromagnetic [15] approaches with a much better spatial and temporal resolution. Faults according to the fault models FM1, FM2, FM3, and FM4 are successively easier to achieve using practical equipment, but they require more elaborate mathematical post-processing and a larger number of injections.

The theoretical analysis is complemented by fault-injection experiments performed by simulation on the hardware level. For the purpose of these experiments, we implemented a circuit for the elliptic curve case, including efficient realizations of scalar multiplication, inversion and division. We simulated the fault injection experiments both in software and in hardware. The outcomes of these experiments confirm the theoretical findings and they show that generating the raw data for the fault attacks in hardware is absolutely practical.

The remainder of the paper is organized as follows. The next section provides an overview of ElGamal-type cryptosystems and includes some other possible application scenarios. Section 3 introduces the four fault models considered in this paper. Section 4 describes the attacks, models them and studies their properties by mathematical means. The software simulations are reported on in Section 5, and the complexity of the necessary fault injections on a hardware platform is discussed in Section 6. Finally, Section 7 concludes the paper.

2 ElGamal-Type Cryptosystems

An ElGamal-type cryptosystem is built from a group G, an element $P \in G$ whose order is a large (prime) number q, a random number $a \in \{2, \ldots, q-2\}$ which serves as the secret key, and the group element $Q = aP$. All information is public except for the secret key a. Notice that we are using the additive notation for G here. This is coherent with the facts that G is usually an abelian group, and that the most interesting case is the group of points of an elliptic curve. Moreover, it is straightforward to change everything to a multiplicatively written group.

Encryption. A message unit is a group element M. To encrypt it, Bob chooses a random integer $k \in \{2, \ldots, q-2\}$, called the *ephemeral key*, and

then calculates the ciphertext unit $C = (C_1, C_2) = (kP,\, M + kQ)$. Thus the ciphertext space is $G \times G$.

Decryption. To decrypt a ciphertext unit $C = (C_1, C_2)$, Alice calculates $C_2 - a\,C_1$ and gets M.

To compute the multiples aP, kP, kQ and aC_1 in this cryptosystem, fast algorithms, usually based on double-and-add techniques, are known (see [41]). Moreover, for the setup phase, there should exist efficient algorithms for computing the order of the group and of an element P in this group. The security of an ElGamal-type cryptosystem rests on the following problem in the group G.

Diffie-Hellman Problem: *Given the two elements $C_1 = kP$ and $Q = aP$ of G, compute the element akP.*

It is quite clear that this problem can be solved if we are able to solve the following stronger problem.

Discrete Logarithm Problem (DLP): *Given the two elements P and $Q = aP$ of G, find a.*

2.1 Some Examples of ElGamal-Type Cryptosystems

Example 2.1. The *classical ElGamal cryptosystem* introduced in [19] uses the multiplicative group $G = (\mathbb{Z}/p\mathbb{Z})^\times$, where p is a large prime. It assumes that there exists a large prime divisor q of $p-1$, for instance if $q = (p-1)/2$ is a prime number. Then we use an element P of G of order q, choose a random number $a \in \{2, 3, \ldots, q-2\}$, and let $Q = P^a$. In this setting, the best currently known algorithms from the index calculus family have a subexponential running time, and the classical ElGamal crytosystem is considered reasonably secure.

Example 2.2. Let p be a (large) prime number, and let $E(\mathbb{F}_p)$ be the set of $\mathbb{F}_p$-rational points of an elliptic curve E over the finite field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. Then the set $E(\mathbb{F}_p)$ is an abelian group. Given a large prime divisor q of the order of $E(\mathbb{F}_p)$ and a point $P \in E(\mathbb{F}_p)$ of order q, we can set up an ElGamal-type cryptosystem by choosing a random number $a \in \{2, \ldots, q-2\}$ and calculating $Q = aP$. This cryptosystem is called the *Elliptic Curve ElGamal (ECEG) cryptosystem*. Its security rests on the Elliptic Curve Diffie-Hellman Problem and the Elliptic Curve Discrete Logarithm Problem. The best current algorithms for these problems have exponential running time, thereby allowing shorter secret keys than for the classical ElGamal cryptosystem.

Example 2.3. In order to find matrix groups with an interesting DLP which cannot be reduced to the case of a finite field, one has to turn to matrix groups over non-commutative rings [38], [29], [30]. Although the security of these cryptosystems against standard attacks has not yet been shown convincingly, they may be attacked successfully using the methods presented in this paper.

2.2 Other Applications

Hitherto, most of the fault attacks on discrete-log-based systems have been applied to signature schemes. Let us briefly explain how.

Remark 2.4. (Signature Schemes) The main problem for using fault attacks against discrete-log-based signature schemes is the use of a random nonce for each signature. This means that the attacker has to be able to verify the same signature again and again, and that the verification step itself is deterministic. Still, if the attacker is able to inject a fault into the secret key during the verification step, simple adaptations of the algorithms given in this paper have been shown to be applicable to the DSA, ECDSA, the Schnorr, and the ElGamal signature schemes (see [4], [16], and [23]). Notice that, almost exclusively, the very restrictive fault model FM1 has been used. Moreover, we point out that countermeasures against disturbing the secret key in signature verifications have been suggested in [25] and [28].

Our final remark indicates how the methods described in this paper can be adapted to factorization based cryptosystems, e.g., to the RSA cryptosystem.

Remark 2.5. (The RSA Cryptosystem) Starting from the very first Bellcore attack [10], most attempts to break the RSA cryptosystem via fault injections targeted the CRT-RSA implementation [32]. Attacks targeting the secret key, as we do in this paper, have been mostly restricted to 1-bit and 2-bit key flips, i.e., to our fault model FM1 and a very strong version of FM2 (see [4], [27] and [5]). Notice that the results presented here can be readily adapted to the RSA cryptosystem by using the group $G = (\mathbb{Z}/n\mathbb{Z})^\times$ and precomputing the values $c^{f_i} \pmod n$, where c is the chosen ciphertext and the numbers f_i correspond to the possible faults. However, due to the long key lengths which are used for RSA cryptosystems nowadays, the fault models FM2, FM3, and FM4 are realistic for small values of s only.

3 Fault Attacks and Fault Models

As for all public-key cryptosystems, a fault attack on an ElGamal-type cryptosystem should target decryption, since only public information is processed during encryption. The attacks described by fault models FM1 through FM4 in this section directly manipulate the bit tuple $\bar{a} = (a_0, \ldots, a_{r-1})$ representing the secret key. These attacks, as mentioned in the introduction, neither modify the group G nor produce elements not contained in G. Therefore, they are not detectable by simple consistency checks. The secret key can be manipulated when it is stored in the memory of the device, when it is processed by the combinational circuitry, or when it is transferred from secure storage (e.g., an EEPROM). Note that we assume that we can perform a decryption without fault injection, repeat it with fault injections, and that the secret key will not change between these decryptions. In the case of FM1, for each key bit, one fault injection affecting it will be sufficient to break the cryptosystem. For the fault models FM2, FM3 and FM4 multiple fault injections affecting each bit are needed in general, again with the same secret key. Let us define the fault models and discuss their physical justification.

Fault Model 1 (FM1): The attack results in a flip of a single bit a_ℓ, where the position $\ell \in \{0, 1, \ldots, r-1\}$ of the bit-flip is unknown to the adversary.

Fault Model 2 (FM2): The attack results in a distortion in an s-bit chunk $(a_\ell, \ldots, a_{\ell+s-1})$ where ℓ is a multiple of a fixed number s. (Here we have $\ell \in \{0, s, \ldots, \lceil r/s \rceil\}$.) The adversary knows which s-bit chunk of $\bar{a}$ is affected but does not know the value of the distortion, i.e., any bit(s) between a_ℓ and $a_{\ell+s-1}$ can be flipped or not, but all bits outside this range are unaffected.

Fault Model 3 (FM3): As in FM2, one s-bit chunk $(a_\ell, \ldots, a_{\ell+s-1})$ with $\ell \in \{0, s, \ldots, \lceil r/s \rceil\}$ is distorted by an unknown value, but the attacker does not know the distorted chunk, i.e., the value of ℓ. Bit flips cannot affect multiple different s-bit chunks.

Fault Model 4 (FM4): The (unknown) distortion again affects one s-bit chunk $(a_\ell, \ldots, a_{\ell+s-1})$, but any arbitrary position ℓ of the chunk within the key is allowed, not necessarily a multiple of s.

FM1 through FM4 span a hierarchy of fault models where the precision achievable by the attacker decreases (and the complexity of the cryptanalysis increases) from FM1 to FM4. Moreover, FM2, FM3 and FM4 include a parameter s, where a larger value of s again stands for a lower attack precision. Note that the requirements of a restrictive fault model can be achieved even using lower-precision equipment by repeating the fault injection multiple times, until the conditions of the fault model happen to apply fortuitously. For example, if the adversary wishes to use single bit-flips according to FM1, but the spatial resolution of the fault-injection equipment can target only chunks of, say, 4 bits that are flipped randomly, she can keep trying fault injections until precisely one of the 4 bits happens to flip. For a system equipped with a detector, this approach increases the probability of successful detection.

FM1 presumes that an attacker can flip a single bit. This may be achievable by high-end laser equipment [1]. In an (unusual) system where the secret key is transferred from secure storage bit-by-bit, the same effect can be reached by applying a disturbance for precisely one cycle during the transfer process. FM2 is applicable to systems which naturally organize information into s-bit chunks, e.g., 8-bit microprocessors [20, 18, 35, 42], 8-bit smartcards [2], 16- and 32-bit systems [22]. Here, a laser or an electromagnetic pulse can target a register where a chunk of the secret key is stored. FM2 also describes an attack against the key being transferred from secure storage in s-bit chunks.

FM3 is the model of choice when an attacker can target one of the registers that hold the secret key, but does not know which chunk is stored in which register. In the context of attacking the key during transfer from secure storage, FM3 can be used if the temporal resolution is poor. This can be the case if the power and electromagnetic traces of the circuit are protected against information leakage and give the attacker no information which chunk of the key is transferred during which clock cycle. Finally, FM4 applies to an implementation where the entire key is stored in a contiguous register of sufficient size and the attack targets a specific memory element of the register, but may hit neighboring memory elements as well, up to a range of s bits in total.

4 Mathematical Description of the Fault Attacks

In the following, using an ElGamal-type cryptosystem, we assume that we have produced a ciphertext point pair (C_1, C_2), where $C_1 = kP$ with an (unknown) random number k, and with $C_2 = M + kQ$, where M is the plaintext unit and $Q = aP$. Moreover, we suppose that we can run the decryption algorithm and compute $M = C_2 - aC_1$ without fault injection, and that we can produce faulty plaintext units $M_i = C_2 - (a + f_i)C_1$ for $i = 1, \ldots, n$, where the integer f_i represents the **i-th fault** and n is the number of fault injections we perform. Then we can calculate

$$M - M_i \;=\; (C_2 - aC_1) - (C_2 - (a + f_i)C_1) \;=\; f_i C_1$$

and note that, since we know C_1, we can precompute various multiples of C_1 and thereby find f_i using a suitable look-up table.

Next we represent the numbers a, k, etc., by bit tuples $\bar{a} = (a_0, \ldots, a_{r-1})$, $\bar{k} = (k_0, \ldots, k_{r-1})$, and so on, where $a_i, k_j \in \{0, 1\}$. In other words, we have $a = a_0 + a_1\, 2 + \cdots + a_{r-1}\, 2^{r-1}$, etc. Here r has to be chosen such that $2^r > \max\{p, q\}$ where q is the order of P. The number r represents the **bit size** of the cryptosystem.

Now we describe the result of the i-th fault injection by the **fault pattern** $\bar{e}_i = (e_{i0}, \ldots, e_{i\,r-1})$, where $e_{ij} \in \{0, 1\}$, and where we have $e_{ij} = 1$ if and only if the j-th bit of $\bar{a}$ is flipped by the i-th fault injection. Hence the bit tuple representing the number $a + f_i$ is given by $(a'_0, \ldots, a'_{r-1})$, where we have $a'_j = a_j$ if $e_{ij} = 0$ and $a'_j = 1 - a_j$ if $e_{ij} = 1$ for $j = 0, \ldots, r-1$. We can write this as $a'_j = e_{ij} + (1 - 2e_{ij})a_j$. Thus the number f_i satisfies

$$f_i = (a'_0 + a'_1 2 + \cdots + a'_{r-1} 2^{r-1}) - (a_0 + a_1 2 + \cdots + a_{r-1} 2^{r-1}) = \sum_{j=0}^{r-1} e_{ij}(1 - 2a_j)\, 2^j$$

and here we have $e_{ij}(1 - 2a_j) \in \{-1, 0, 1\}$. Observe that this yields the bounds $-2^r + 1 \le f_i \le 2^r - 1$. In view of this representation, the possible fault patterns leading to the observation of f_i depend on the following notion.

Definition 4.1. Let $f_i \in \mathbb{Z}$ be such that $-2^r + 1 \le f_i \le 2^r - 1$. Then a tuple $B = (b_0, \ldots, b_{r-1}) \in \{-1, 0, 1\}^r$ which satisfies $f_i = b_0 + 2b_1 + \cdots + 2^{r-1} b_{r-1}$ is called a **signed r-bit representation** of f_i.

Unfortunately, the signed r-bit representation of a number f_i is, in general, not uniquely determined. For instance, Figure 1 plots the number of signed 8-bit representations for the integers in the range $\{-255, -254, \ldots, 255\}$.

Let $\lambda(f_i, r)$ denote the number of signed r-bit representations of f_i. The function $\lambda(f_i, r)$ was studied in [17]. In particular, Lemmas 3 and 4 of this paper provide recursive formulas which allow us to compute the values of $\lambda(f_i, r)$ easily.

For a fixed signed r-bit representation $B = (b_0, \ldots, b_{r-1})$ of f_i, the positions where $b_j \in \{1, -1\}$ determine both the fault pattern and the values of a_j at these positions, since $b_j = e_{ij}(1 - 2a_j) \neq 0$ iff $e_{ij} = 1$, and in this case $a_j = (1 - b_j)/2$. Thus our method of analyzing the fault injections proceeds via the following steps:

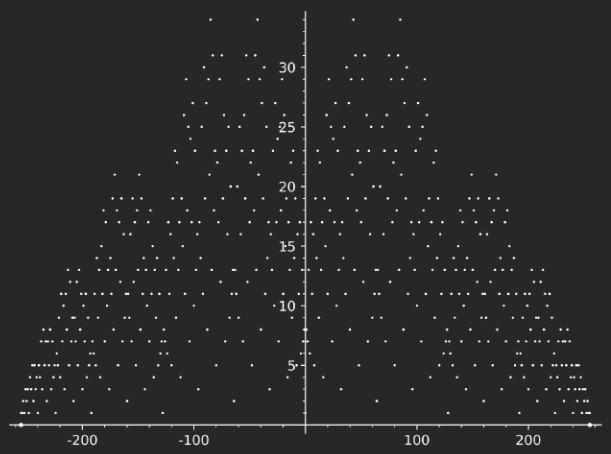

Fig. 1. Number of signed 8-bit representations

(1) From the observed element M_i and the list of precomputed multiples of C_1, determine the number $f_i \in \mathbb{Z}$.
(2) Find all signed r-bit representations of f_i which are compatible with the given fault model.
(3) For each of these, obtain the set of secret keys a compatible with it, and deduce the *key candidate set* consisting of all secret keys a which are compatible with the observation of M_i.
(4) Repeat the fault injections until the key candidate set is small enough for an exhaustive search.

Next we work out the details for the fault models FM1, FM2, FM3, and FM4.

4.1 The Attack for the Fault Model FM1

By assumption, the i-th fault pattern is given by $\bar{e}_i = (e_{i0}, \ldots, e_{i\,r-1})$, where $e_{i\nu(i)} = 1$ for a randomly chosen position $\nu(i) \in \{0, \ldots, r-1\}$, and where $e_{ij} = 0$ otherwise. Now the i-th fault f_i determines exactly one bit of a as follows.

Proposition 4.2. *For the fault model FM1, the following claims hold.*

(a) For the above fault pattern $\bar{e}_i$, we have $f_i \in \{-2^{\nu(i)}, 2^{\nu(i)}\}$. Consequently, we obtain $a_{\nu(i)} = 0$ if $f_i = 2^{\nu(i)}$ and $a_{\nu(i)} = 1$ if $f_i = -2^{\nu(i)}$.
(b) The expected number of successful fault injections needed to recover the full secret key a is $r\,(1 + \frac{1}{2} + \cdots + \frac{1}{r}) \approx r\,\ln(r)$.

Proof. To prove (a), we note that there is only one position j with $e_{ij} \neq 0$, namely $j = \nu(i)$. Hence the signed r-bit representation of f_i has only one non-zero digit, namely the $\nu(i)$-th digit. Together with the above discussion, the claims follow.

For the proof of (b), we note that, after having found k bits of the secret key, the probability of discovering a new bit using one fault injection is $\frac{r-k}{r}$. Repeating injections until a new bit is affected is distributed geometrically, and thus has an expected value of $\frac{r}{r-k}$ injections. Altogether, we expect $\frac{r}{r} + \frac{r}{r-1} + \cdots + \frac{r}{1}$ injections until we have found the full secret key. □

This proposition leads us immediately to Algorithm 1.

Algorithm 1 (FM1 Fault Analysis Algorithm)

Input: A ciphertext pair $C = (C_1, C_2) \in G \times G$, the correct plaintext unit M and faulty plaintext units M_i for $i = 1, \ldots, n$.
Output: A bittuple $(a_0, \ldots, a_{r-1})$ representing the secret key a.

$L_1 := \{2^i C_1 \mid i = 0, \ldots, r-1\}$;
$L_2 := \{-2^i C_1 \mid i = 0, \ldots, r-1\}$;
$(a_0, \ldots, a_{r-1}) := (2, \ldots, 2)$;
for $i = 1$ **to** n **do**
  if $M - M_i = 2^{\nu(i)} C_1 \in L_1$ **then**
    $a_{\nu(i)} := 0$;
  else if $M - M_i = -2^{\nu(i)} C_1 \in L_2$ **then**
    $a_{\nu(i)} := 1$;
  end if
end for
if $(a_j \in \{0,1\}$ **for** $j = 0, \ldots, r-1)$ **then**
  return $(a_0, \ldots, a_{r-1})$
else
  return "not enough injections"
end if

Clearly, this algorithm requires the calculation of $r-1$ doublings and $n \approx r\,\ln(r)$ additions in the group G. A further speed-up can be achieved by choosing the neutral element of G for M, in which case no additions are necessary.

4.2 The Attack for the Fault Model FM2

In this section we continue to assume that the numbers a, k, etc., are represented by r-bit tuples $\bar{a} = (a_0, \ldots, a_{r-1})$, etc. In FM2, we attack a chosen fixed s-bit subtuple of a. Let the s-bit subtuple under attack be given by $\bar{a}^{(\ell)} = (a_\ell, a_{\ell+1}, \ldots, a_{\ell+s-1})$, where $\ell \in \{0, \ldots, r-s\}$. Correspondingly, the only part of the i-th fault pattern which may be non-zero is $\bar{e}_i^{(\ell)} = (e_{i\,\ell}, \ldots, e_{i\,\ell+s-1}) \in \{0,1\}^s$. Hence we may write the i-th fault f_i in the form

$$f_i \;=\; \sum_{j=\ell}^{\ell+s-1} e_{ij}\,(1-2a_j)\,2^j \;=\; 2^\ell \sum_{j=0}^{s-1} \tilde{e}_{ij}\,(1-2\tilde{a}_j) \;=:\; 2^\ell\,\tilde{f}_i$$

where $\tilde{e}_{ij} = e_{i\,j+\ell}$, where $\tilde{a}_j = a_{j+\ell}$, and where $\tilde{f}_i$ is given by the second sum. Recall that we have to examine the signed s-bit representations of f_i. Since

$f_i = 2^\ell \tilde{f}_i$, it clearly suffices to look at the signed s-bit representations of $\tilde{f}_i$. They allow us to construct the following sets.

Definition 4.3. Given a signed s-bit representation $B = (b_0, \ldots, b_{s-1})$ of $\tilde{f}_i$, we let $K_B(\tilde{f}_i)$ be the set of all s-bit tuples $(\tilde{a}_0, \ldots, \tilde{a}_{s-1})$ such that $\tilde{a}_j = (1-b_j)/2$ whenever $b_j \neq 0$. Then the union of all sets $K_B(\tilde{f}_i)$, where B traverses the signed s-bit representations of $\tilde{f}_i$, is called the **key subtuple candidate set** of $\tilde{f}_i$ and is denoted by $K(\tilde{f}_i)$.

Notice that we have $-2^s + 1 \leq \tilde{f}_i \leq 2^s - 1$. By evaluating all 3^s signed s-bit tuples, we could precompute for each number $\tilde{f}_i$ the set of all signed s-bit representations of $\tilde{f}_i$. Then, by taking the union of all these sets, we could precompute all key subtuple candidate sets $K(\tilde{f}_i)$ for $\tilde{f}_i \in \{-2^s+1, \ldots, 2^s-1\}$. Figure 2 illustrates the number of key subtuple candidates as a function of $\tilde{f}_i$ in the case $s = 8$.

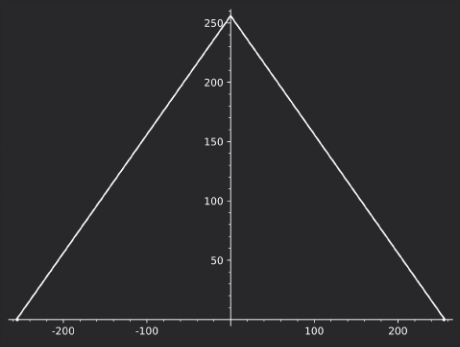

Fig. 2. Number of key byte candidates

However, this precomputation is in reality not necessary, since the key subtuple candidate sets can be described explicitly, as the next proposition shows.

Proposition 4.4. *Let $\tilde{f}_i \in \{-2^s+1, -2^s+2, \ldots, 2^s-1\}$. Then the key subtuple candidate set $K(\tilde{f}_i)$ is given by*

$$K(\tilde{f}_i) \;=\; \{c \in \{0,1\}^s \mid 0 \leq \tilde{f}_i + N(c) \leq 2^s - 1\}.$$

where $N(c) = c_0 + 2c_1 + \cdots + 2^{s-1}c_{s-1}$ for $c = (c_0, \ldots, c_{s-1}) \in \{0,1\}^s$.

Proof. First we prove the inclusion $\subseteq$. Given $c \in K(\tilde{f}_i)$, there exists a signed s-bit representation $B = (b_0, \ldots, b_{s-1})$ of $\tilde{f}_i$ such that $c \in K_B(\tilde{f}_i)$. Writing $c = (c_0, \ldots, c_{s-1})$, we have $N(c) = c_0 + 2c_1 + \cdots + 2^{s-1}c_{s-1}$ and $c_j = (1-b_j)/2$ for all j with $b_j \neq 0$. Therefore we obtain

$$\tilde{f}_i + N(c) \;=\; \sum_{\{j \mid b_j \neq 0\}} (b_j + (1-b_j)/2)\, 2^j \;+\; \sum_{\{j \mid b_j = 0\}} c_j\, 2^j$$

and then $b_j + (1 - b_j)/2 = (1 + b_j)/2 \in \{0, 1\}$ leads to $0 \le \tilde{f}_i + N(c) \le 2^s - 1$.

It remains to prove the inclusion $\supseteq$. For $\tilde{f}_i = 0$, we may use the signed s-bit representation $B = (0, \ldots, 0)$, since in this case all s-bit tuples c are in $K_B(\tilde{f}_i)$. Next we consider the case $\tilde{f}_i \neq 0$. Let $c = (c_0, \ldots, c_{s-1})$ be a bit tuple such that $\tilde{c} := \tilde{f}_i + N(c) \in \{0, \ldots, 2^s - 1\}$. Then the number $\tilde{c}$ has a binary representation $\tilde{c} = \tilde{c}_0 + \tilde{c}_1\, 2 + \cdots + \tilde{c}_{s-1} 2^{s-1}$, where $\tilde{c}_i \in \{0, 1\}$. For $j = 0, \ldots, s - 1$, we define

$$b_j = \begin{cases} 1 & \text{if } c_j = 0 \text{ and } \tilde{c}_j = 1, \\ -1 & \text{if } c_j = 1 \text{ and } \tilde{c}_j = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, the componentwise sum of the tuples $(b_0, \ldots, b_{s-1})$ and $(c_0, \ldots, c_{s-1})$ equals $(\tilde{c}_0, \ldots, \tilde{c}_{s-1})$. Therefore we have $\sum_{j=0}^{s-1} b_j\, 2^j = \tilde{c} - N(c) = \tilde{f}_i$, which means that $B = (b_0, \ldots, b_{s-1})$ is a signed s-bit representation of $\tilde{f}_i$. Furthermore, since $c_j = (1 - b_j)/2$ for $j = 0, \ldots, s - 1$, we have $c \in K_B(\tilde{f}_i)$, and the proof is complete. □

Based on this proposition, we obtain the FM2 Fault Analysis Algorithm described below.

Algorithm 2 (FM2 Fault Analysis Algorithm)

Input: A ciphertext pair $C = (C_1, C_2) \in G \times G$, the correct plaintext point M, faulty plaintext points M_i for $i = 1, \ldots, n$, and a number $\ell \in \{0, \ldots, r - s\}$.

Output: The s-bit tuple $a^{(\ell)} = (a_\ell, \ldots, a_{\ell+s-1})$ representing the s-bit subtuple of the secret key a starting at index ℓ.

1: $L := \{j\, 2^\ell\, C_1 \mid j = -2^s + 1, \ldots, 2^s - 1\}$;
2: $K_{\min} := 0$; $K_{\max} := 2^s - 1$;
3: **for** $i = 1$ **to** n **do**
4: Find $\tilde{f}_i \in \{-2^s + 1, \ldots, 2^s - 1\}$ such that $M - M_i = \tilde{f}_i\, 2^\ell\, C_1 \in L$.
5: $K_{\min} := \max\{K_{\min}, -\tilde{f}_i\}$; $K_{\max} := \min\{K_{\max}, 2^s - 1 - \tilde{f}_i\}$;
6: **if** $K_{\min} = K_{\max}$ **then**
7: **return** the s-bit binary representation $a^{(\ell)} \in \{0, 1\}^s$ of $K_{\min}$.
8: **end if**
9: **end for**
10: **return** "not enough injections"

For every fault injection, only one addition in G and some easy fixed-cost operations are necessary. By choosing the neutral element of G for M, we can even get rid of the addition and reduce the running time of the algorithm to almost zero. Notice that Step (5) uses Proposition 4.4 to improve the lower and upper bounds for the numbers representing the tuples which satisfy the inequalities implied by the i-th fault injection. The number of necessary fault injections until only one key candidate remained was experimentally found to be around $n \approx 1.5 \cdot 2^s$ for small s. More detailed values for different values of s are

given in the next section. Of course, since we are actually interested in the full secret key, we have to repeat this attack $\lceil \frac{r}{s} \rceil$ times, so that the full key recovery requires approximately $1.5 \cdot 2^s \cdot \lceil \frac{r}{s} \rceil$ fault injections.

4.3 The Attack for the Fault Model FM3

For the fault model FM3, we assume that $r = st$ for some $t \geq 1$ and that the i-th fault pattern is of the form $\bar{e}_i^{(\ell)} = (e_{i\,\ell}, \ldots, e_{i\,\ell+s-1})$, where $\ell \in \{0, s, 2s, \ldots, (t-1)s\}$. Note that, for simplicity, we suppose that r is divisible by s. For the observed i-th fault f_i, we observe that it is of the form $f_i = \sum_{j=\ell}^{\ell+s-1} e_{ij}\,(1 - 2a_j)\,2^j$, and therefore it satisfies $2^\ell \leq f_i \leq 2^{\ell+s} - 1\}$. Consequently, a non-zero fault f_i implies that we can read off the value of ℓ via $\ell = \max\{j \in \{0, \ldots, t-1\} \mid 2^{sj} \text{ divides } f_i\}$. Hence the FM3 Fault Analysis Algorithm can be derived from the FM4 version in a straightforward manner.

Algorithm 3 (FM3 Fault Analysis Algorithm)

Input: A ciphertext pair $C = (C_1, C_2) \in G \times G$, the correct plaintext point M, positive numbers s, t, and faulty plaintext points M_i for $i = 1, \ldots, n$.
Output: A bittuple $(a_0, \ldots, a_{r-1})$ representing the secret key a.

for $\ell = 0$ **to** $t-1$ **do**
  $L_\ell := \{j\,2^{\ell s}\,C_1 \mid j = -2^s + 1, \ldots, 2^s - 1\}$;
  $K_{\min}^{(\ell)} := 0$; $K_{\max}^{(\ell)} := 2^s - 1$;
end for
for $i = 1$ **to** n **do**
  Compute $M - M_i$ and find $\ell \in \{0, \ldots, t-1\}$ such that $M - M_i \in L_\ell$.
  Find $\tilde{f}_i \in \{-2^s + 1, \ldots, 2^s - 1\}$ such that $M - M_i = \tilde{f}_i\,2^{\ell s}\,C_1 \in L_\ell$.
  $K_{\min}^{(\ell)} := \max\{K_{\min}^{(\ell)}, -\tilde{f}_i\}$; $K_{\max}^{(\ell)} := \min\{K_{\max}^{(\ell)}, 2^s - 1 - \tilde{f}_i\}$;
  if $(K_{\min}^{(m)} = K_{\max}^{(m)}$ **for** $m = 0, \ldots, t-1)$ **then**
    for $m = 0, \ldots, t-1$ **do**
      Let $a^{(m)}$ be the s-bit binary representation of $K_{\min}^{(m)}$.
    end for
    return the concatenation of the tuples $a^{(0)}, \ldots, a^{(t-1)}$.
  end if
end for
return "not enough injections"

If the fault injections affected the different s-bit subtuples of the key independently at random, we can get a rough estimate for the average number of necessary injections by multiplying the corresponding value from the FM2 attack by $t\,\ln(t)$. A more detailed experimentally determined set of values is given in the next section.

4.4 The Attack for the Fault Model FM4

The fault model FM4 implies that the i-th error vector is of the form $\bar{e}_i = (e_{i0}, \ldots, e_{i\,r-1})$, where $e_{ij} = 0$ for $j \notin \{\ell_i, \ell_i + 1, \ldots, \ell_i + s - 1\}$ with some (unknown) number $\ell_i \in \{0, \ldots, r - s\}$. Therefore the i-th fault f_i is of the form $f_i = \sum_{j=\ell_i}^{\ell_i+s-1} e_{ij}(1 - 2a_j)\, 2^j$. Given the value of f_i, we can determine the possible values of ℓ_i as follows.

Proposition 4.5. *Let $s \geq 1$, let $f_i \in \{-2^r + 1, \ldots, 2^r - 1\} \setminus \{0\}$, and let ℓ_i be defined as above. Moreover, let $f_i = b_0 + b_1 2 + \cdots + b_{r-1} 2^{r-1}$ be a signed r-bit representation of f_i, where $b_j \in \{-1, 0, 1\}$.*

(a) Let $\mu_i = \max\{j \geq 0 \mid 2^j$ divides $f_i\}$. Then every signed r-bit representation of f_i starts in the $(\mu_i{+}1)$-st position, i.e., we have $b_j = 0$ for $j < \mu_i$ and $b_{\mu_i} \neq 0$.
(b) Write $f_i = 2^{\mu_i} \tilde{f}_i$, and let $|\tilde{f}_i| = \varepsilon_0 + \varepsilon_1 2 + \cdots + \varepsilon_k 2^k$ be the binary representation of the absolute value of $\tilde{f}_i$, where $\varepsilon_j \in \{0, 1\}$ for $j = 0, \ldots, k-1$ and $\varepsilon_k = 1$. Then every signed r-bit representation of f_i satisfies the inequality $\max\{j \geq 0 \mid b_j \neq 0\} \geq \mu_i + k$.
(c) For the number ℓ_i, we have $\ell_i \in \{\mu_i + k - s + 1, \ldots, \mu_i\}$.

Proof. First we prove (a). By hypothesis, the number f_i is divisible by 2^{μ_i}. Hence the number $b_0 + b_1 2 + \cdots + b_{\mu_i - 1} 2^{\mu_i - 1}$ is divisible by 2^j for $j = 1, \ldots, \mu_i$. Induction on j shows $b_0 = \cdots = b_{\mu_i - 1} = 0$. On the other hand, if $b_{\mu_i} = 0$ then f_i is even divisible by $2^{\mu_i + 1}$, in contradiction to the definition of μ_i.

Next we show (b). Let us assume $f_i > 0$. In the case $f_i < 0$, the proof proceeds analogously. It suffices to show that every signed binary representation $\tilde{f}_i = \tilde{b}_0 + \cdots + \tilde{b}_m 2^m$ with $\tilde{b}_j \in \{-1, 0, 1\}$ and $\tilde{b}_m \neq 0$ satisfies $m \geq k$. Suppose that $m < k$. Then $\tilde{b}_0 + \cdots + \tilde{b}_m 2^m \leq 1 + 2 + \cdots + 2^m < 2^k \leq \tilde{f}_i$ yields a contradiction.

Finally, we prove (c). By (b), the last non-zero digit b_ν of any signed r-bit representation of f_i satisfies $\nu \geq \mu_i + k$. Since the corresponding fault pattern has at most s non-zero signed digits, the minimal index of the first signed digit of f_i inside the fault pattern is $\mu_i + k - s + 1$. By (a), the maximal value of ℓ_i is μ_i, because this index is certainly inside the fault pattern. □

Notice that in part (b) of the proposition we also have $k \leq s - 1$, since the fault pattern consists of s consecutive bits and b_k is still inside this region. Thus we define $\delta = s - 1 - k \geq 0$. In view of the possible values of ℓ_i determined by part (c) of the proposition, the maximal range of signed binary digits of f_i affected by the fault injection, and therefore the maximal range of non-zero signed binary digits, is $(b_{\mu_i - \delta}, b_{\mu_i - \delta + 1}, \ldots, b_{\mu_i + s - 1})$. However, by the definition of μ_i, the signed binary digits $b_{\mu_i - \delta}, \ldots, b_{\mu_i - 1}$ of f_i are always zero. Hence the range of indices for which we can hope to restrict the set of key subtuple candidates is $(\mu_i, \ldots, \mu_i - s + 1)$. Let us introduce this set of key subtuple candidates formally.

Definition 4.6. Given an FM4 fault $f_i \in \{-2^r+1, \ldots, 2^r-1\} \setminus \{0\}$, we define $\mu_i = \max\{j \geq 0 \mid 2^j \text{ divides } f_i\}$. Then the set

$$K(f_i) \;=\; \{(c_0, \ldots, c_{r-1}) \in \{0,1\}^r \mid f_i = \sum_{j=\mu_i}^{\mu_i+s-1} e_{ij}(1-2c_j)\,2^j \text{ for some fault pattern } (e_{i0}, \ldots, e_{i\,r-1})\}$$

is called the **key subtuple candidate set** of f_i.

Given f_i, the key subtuple candidate set can be found using the following proposition.

Proposition 4.7. *Let $f_i \in \{-2^r+1, \ldots, 2^r-1\} \setminus \{0\}$ be the fault returned by an FM4 fault injection, and let $\mu_i = \max\{j \geq 0 \mid 2^j$ divides $f_i\}$. Then we have*

$$K(f_i) \;=\; \{c \in \{0,1\}^r \mid 0 \leq 2^{-\mu_i} f_i + N_{\mu_i}(c) \leq 2^s - 1\}$$

where $N_{\mu_i}(c) = c_{\mu_i} + c_{\mu_i+1}\,2 + \cdots + c_{\mu_i+s-1}\,2^{s-1}$.

Proof. By the definition of μ_i, we know that $b_{\mu_i} \neq 0$ for every signed r-bit representation $(b_0, \ldots, b_{r-1})$ of f_i. The inequalities follow from Proposition 4.4 and the above discussion.

The restrictions on the possible secrets keys provided by FM4 fault injections affect a subtuple of the secret key, where the position of this subtuple depends on the observed fault f_i. When we perform an injection sequence, we may combine the various restrictions in one of the following ways:

(I) After writing down all linear inequalities for the indeterminates $c_0, \ldots, c_{r-1}$ resulting from the bounds in Proposition 4.7, we search for an element in $\{0,1\}^r$ which satisfies all these inequalitites. To combine the bounds for each range of indices $(\ell, \ldots, \ell+s-1)$ where $\ell \in \{0, \ldots, r-s\}$, we have to use some version of interval logic, for instance the Allen calculus. Then we inject faults until the number of remaining key candidates is small enough for exhaustive search.

(II) For every observed fault f_i, we construct all signed r-digit representations whose outermost non-zero digits are at most $s-1$ digits apart. For each of these representations, we write down the implications on the bits of the secret key it implies as a predicate logic formula involving $c_0, \ldots, c_{r-1}$. Then we use a SAT solver to deal with the resulting set of formulas.

For the purposes of this paper, we implemented the second method. Let us work through an example to make the process more explicit.

Example 4.8. In the case $r = 4$ and $s = 2$, suppose that the secret key is $a = (0,1,0,0)$. We inject FM4 faults using various randomly chosen error patterns and compute the restrictions on the key candidates (c_0, c_1, c_2, c_3) they yield.

Algorithm 4 (FM4 Fault Analysis Algorithm)

Input: A ciphertext pair $C = (C_1, C_2) \in E(\mathbb{F}_p)^2$, the correct plaintext point M and faulty plaintext points M_i for $i = 1, \ldots, n$.
Output: The r-bit tuple $(a_0, \ldots, a_{r-1})$ representing the secret key a.

```
Construct the set E of all possible fault patterns.
for all ē ∈ E do
    Construct the set F_ē of all faults for the pattern ē.
end for
Let L be the set of all fC_1 with f ∈ F_ē for some ē ∈ E.
for i = 1 to n do
    Find f_i ∈ {−2^r + 1, ..., 2^r − 1} such that M − M_i = f_i C_1 ∈ L.
    Compute the set B of all signed binary representations b̄ = (b_0, ..., b_{r−1}) of f_i
    of width(b̄) ≤ s.
    for all (b_0, ..., b_{r−1}) ∈ B do
        Consider the set S consisting of all literals c_j for which b_j = 1 and all literals
        ¬c_j for which b_j = −1.
        Let C_b̄ be the conjunction of the literals in S.
    end for
    Let D_i be the disjunction of all formulas C_b̄ with b̄ ∈ B.
end for
Compute the set K of all satisfying assignments for D_1, ..., D_n.
if #K = 1 then
    return the element (a_0, ..., a_{r−1}) of K.
else
    return "not enough injections"
end if
```

(1) For $\bar{e}_1 = (1, 0, 0, 0)$, the fault $f_1 = 1$ is returned. Two signed binary representations exist for f_1, namely $(1, 0, 0, 0)$ and $(-1, 1, 0, 0)$. They imply $(c_0 = 0) \vee (c_0 = 1 \wedge c_1 = 0)$.
(2) For $\bar{e}_2 = (0, 0, 1, 1)$, the fault $f_2 = 12$ is returned. It has only one signed binary representation $(0, 0, 1, 1)$ and implies $c_3 = c_4 = 0$.
(3) For $\bar{e}_3 = (0, 1, 0, 0)$, the fault $f_3 = -2$ is returned. Its signed binary representations $(0, -1, 0, 0)$ and $(0, 1, -1, 0)$ imply $(c_1 = 1) \vee (c_1 = 0 \wedge c_2 = 1)$.

These three fault injections reduce the key candidate set already to $\{(0, 1, 0, 0)\}$, and the attack is complete.

Consequently, Method (II) above can used to construct the following fault analysis algorithm for the fault model FM4. Here we let the **width** of a signed binary representation $\bar{b} = (b_0, \ldots, b_{r-1})$ be given by $\text{width}(\bar{b}) = \nu - \mu + 1$, where $\mu = \min\{j \mid b_j \neq 0\}$ and $\nu = \max\{j \mid b_j \neq 0\}$.

Clearly, this algorithm becomes cumbersome for large values of s. In this case, Method (I) yields a better algorithm. The average number of fault injections needed for various values of the pair (r, s) is determined experimentally in the next section.

5 Software Simulations

The following data were obtained with a prototype implementation of the proposed fault attacks and fault analysis algorithms which uses the interpreted top-level language of the computer algebra system ApCoCoA (cf. [3]). Since the mathematical components of the attacks are very efficient, the computational power of a laptop having an Intel i5-4300M microprocessor and 16 GB of RAM was fully sufficient.

Simulating Fault Model FM1 Both the generation of a randomly located bit flip in the bit-tuple $(a_0, \ldots, a_{r-1})$ representing the secret key a and the recovery of the corresponding key bit from the faulty point M_i take only a fraction of a second. Also the pre-computation of the lists L_1 and L_2 is a matter of a few seconds, even in the case of the largest curves under consideration.

For the various values of r, the following NIST curves were used:

(1) The curve *secp128r1* is given in [13].
(2) The curves *secp192r1*, *secp224r1*, *secp256r1*, and *secp384r1* are also given in [13]. They are called P-192, P-224, P-256, and P-385, resp., in NIST FIPS 186-4 (see [39]) and recommended for official usage there.

Table 1 collects the experimentally found values for the number of necessary fault injections. In each case, we performed 500 sequences of simulated injections, each lasting until the secret a was fully recovered. We list the minimum and maximum number of injections needed, called $N_{\min}$ and $N_{\max}$, the average number N_{avg} of injections needed, as well as the average time T_{avg} to recover the secret key from one injection sequence.

Table 1. Number of injections for fault model FM1

curve	$N_{\min}$	$N_{\max}$	N_{avg}	T_{avg} (sec)
secp128r1	373	1270	670	0.092
secp192r1	624	2657	1129	0.143
secp224r1	751	3022	1352	0.179
secp256r1	856	3360	1589	0.212
secp384r1	1589	5001	2487	0.368

Note that the number N_{avg} approximates the theoretical value $r(1 + \frac{1}{2} + \cdots + \frac{1}{r})$ very well in each case, but that the standard deviation is rather large.

Simulating Fault Model FM2 For the fault model FM2, we assume that we are able to inject a random fault into an s-bit subtuple of the secret key. Thus a fault is an s-bit tuple which characterizes the positions of the bit flips in the given subtuple $(a_\ell, \ldots, a_{\ell+s-1})$ of the secret key. For the software simulation, we

suppose that the fault pattern subtuples $\bar{e}_i^{(\ell)} = (e_{i\,\ell}, \ldots, e_{i\,\ell+s-1})$ are equidistributed in the set of all s-bit tuples, where $e_{ij} = 1$ if and only if a_j has been flipped.

Then we let $b_{\ell+j} = e_{\ell+j}(1 - 2a_{\ell+j})$ for $j = 0, \ldots, s-1$ and note that the resulting number $\tilde{f}_i = b_\ell + \cdots + 2^{s-1} b_{\ell+s-1}$, which is observed via the look-up table for $M - M_i = \tilde{f}_i\, C_1^{(\ell)}$, is not equidistributed anymore in the range $\{-2^s + 1, -2^s + 2, \ldots, 2^s - 1\}$. According to Algorithm 2, we repeat such fault injections and intersect the sets $K(\tilde{f}_i)$ until the remaining key candidate set has only one element. This leads to the natural question how many fault injections are needed on average.

Table 2 lists the results of the following experiment: we inject faults into a fixed s-bit segment of a, where $s \geq 3$. Depending on s, i.e., depending on our spacial resolution, we tabulate the average number of injections N_{avg} needed. Here we used 20 different random values of a and averaged over 1000 fault injection sequences for each a.

Table 2. Number of injections for the s-bit version of FM2

s	3	4	5	6	7	8
N_{avg}	8.99	16.24	35.04	91.86	155.96	381.88

In the case $s = 8$, our experimental results agree very well with the theoretically derived $N_{\text{avg}} = 381.5$ given in [24]. Since we have equality in Proposition 4.4, whereas [24] merely seems to use the containment $\subseteq$, we can conclude that these bounds represent the best possible values. Notice that the average number of injections needed depends on the actual value of the s-bit subtuple $\bar{a}^{(\ell)} = (a_\ell, \ldots, a_{\ell+s-1})$ of the secret key. For instance, in the case $s = 3$, these average numbers are distributed as given in Table 3.

Table 3. Number of 3-bit injections per secret key

$a^{(\ell)}$	(0,0,0)	(1,0,0)	(0,1,0)	(1,1,0)	(0,0,1)	(1,0,1)	(0,1,1)	(1,1,1)
N_{avg}	7.54	12.01	12.29	11.81	11.81	12.29	12.01	7.54

This table is unchanged if we replace (a_0, a_1, a_2) by $(1 - a_0, 1 - a_1, 1 - a_2)$, since the probabilities of a bit flip from 0 to 1 and from 1 to 0 are identical in our setting.

Simulating Fault Model FM3 The average number of injections needed to recover a secret r-bit key a can be estimated by requiring that at least N_{avg} random faults are injected into each of the $\lceil r/s \rceil$ subtuples of a. However, we prefer to perform experimental determination of these numbers for various sizes

of r and s. Trying 100 fault injection sequences for each of 20 randomly chosen secret keys yields the following table.

Table 4. Average number of injections needed for some versions of FM3

(r,s)	$(16,2)$	$(16,3)$	$(16,4)$	$(16,5)$	$(16,6)$	$(128,2)$	$(128,3)$	$(128,4)$	$(128,5)$
N_{avg}	59.1	83.7	134.2	207.9	521.3	692.7	1203	2018	2615

In the next section we will see that these numbers of injections can be achieved on modern hardware without difficulty.

Simulating Fault Model FM4 Similar to the case of FM3, also for the fault model FM4 it is possible to work out formulas for the numbers of fault injections needed on average to discover the secret key a completely. However, since the number of necessary fault injections increases rapidly, it may be practically more efficient to reduce the key space to a manageable number, say 2^{30} key candidates, and perform an exhaustive search on them. The following table gives some average numbers of fault injections needed to determine a completely in the cases $r = 16$ and $r = 128$.

Table 5. Average number of injections needed for FM4

(r,s)	(16,2)	(16,3)	(16,4)	(16,5)	(128,2)	(128,3)	(128,4)
N_{avg}	65.6	95.2	202.7	334.1	887	1831	3180

6 Fault Injections in Hardware

To assess the complexity of actually performing the fault attacks according to the models FM1–FM4, we ran several fault injection experiments. To this end, we used a circuit implementation of the elliptic curve (ECEG) cryptosystem with curve *secp128r1* (see [13]), designed using the *Xilinx Vivado Design Suite*. The circuit included a point multiplication module, an encryption module, and a decryption module. The point multiplication module was used to generate the public point $Q = aP$. The result, together with the generating point P, forms the input of the encryption module, which generates the encrypted message pair (C_1, C_2). The encrypted message pair and the secret key a are then decrypted to output the message M. The secret key is disturbed, i.e., the XOR difference between the fault-free and the faulty secret keys is a random fault $\bar{e}_i$, and the fault-affected decrypted message M_i is recorded. These messages are handed over to the mathematical analysis.

Notice that fault model FM4 specializes to FM1 if we use $s = 1$. For the fault models FM2 and FM3, we attacked only a single s-bit subtuple of the key

Table 6. Hardware Timings for the curve *secp128r1*

FM2 and **FM3**			**FM4** and **FM1** (for $s = 1$)		
s	n	clock cycles	s	n	clock cycles
3	13	22,438,196	1	700	1,663,035,288
4	25	43,535,816	2	1,000	2,372,216,772
5	58	85,406,538	3	1,500	3,552,932,556
6	116	167,017,554	4	2,500	5,918,604,705
7	230	326,215,109	5	6,500	15,374,014,130
8	450	633,178,049	6	13,000	30,740,627,180

bit tuple $\bar{a}$, namely the least significant subtuple $\bar{a}^{(0)} = (a_0, \ldots, a_{s-1})$. This is clearly sufficient for estimating the complexity of hardware fault injection.

Table 6 contains the total number of clock cycles needed for n fault injections, where n is selected such as to obtain a single or very few key subtuple candidates. Note that some of the fault injections may result in the same $\bar{e}_i$ being injected, in which case the n faulty points will include duplicates, and multiple key subtuple candidates may result. (This case did occur in our experiments).

For FM2, this attack must be applied to $t = \lceil r/s \rceil$ different s-bit key subtuples, in order to recover the full secret key. Hence the numbers in the table have to be multiplied by t for a full key recovery. For FM3, the numbers have to be multiplied by $t \ln(t)$, since we have to hit each key subtuple the given number of times.

According to Table 6, the attacks are conducted within 31 billion clock cycles, or roughly 26 minutes on a low-cost 20 MHz device. This demonstrates that the raw data needed for the mathematical analysis of the previous sections can be collected by an adversary within a reasonable amount of time.

7 Conclusions

In this paper we proposed a hierarchy of four fault models for ElGamal-type cryptosystems. They allow the adversary to trade the precision of the fault-injection equipment for the number of necessary fault injections and the complexity of the resulting mathematical analysis. While these fault models are applicable to all ElGamal-type cryptosystems, we studied specific attacks on ECEG cryptosystems, where they can be seen as generalizations of the previously used bit and byte faults. We systematically investigated the relationship between the size of the cryptosystem, the parameter s that describes the fault injection accuracy, and the number of necessary fault injections. We also reported on the effort required in hardware to produce the physical raw data for the attacks.

In the future, we plan to investigate whether attacks on ElGamal-type cryptosystems can be constructed automatically. An automatic procedure would be given only the cipher description and the fault model as inputs, and it would produce the mathematical analysis similar to Fault Analysis Algorithms 1–4 in this paper.

Acknowledgements

This research was supported by DFG (German Research Foundation) project "Algebraische Fehlerangriffe" grants PO 1220/7-2 and KR 1907/6-2.

References

1. Agoyan, M., Dutertre, J.M., Mirbaha, A.P., Naccache, D., Ribotta, A.L., Tria, A.: How to flip a bit? In: 16th Int. On-Line Testing Symposium (IOLTS). pp. 235–239. IEEE, Piscataway, NJ, USA (2010)
2. Aigner, H., Bock, H., Hütter, M., Wolkerstorfer, J.: A low-cost ecc coprocessor for smartcards. In: Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 107–118. LNCS 3156, Springer-Verlag, Berlin, Heidelberg (2004)
3. ApCoCoA Team, T.: ApCoCoA: Applied Computations in Computer Algebra (2013), http://apcocoa.uni-passau.de
4. Bao, F., Deng, R., Han, Y., Jeng, A., Narasimhalu, A., Ngair, T.H.: Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. In: Christianson, B., Crispo, B., Lomas, T., Roe, M. (eds.) 5th Int. Workshop on Security Protocols. pp. 115–124. LNCS 1361, Springer-Verlag, Berlin, Heidelberg (1997). https://doi.org/10.1007/BFb0028164
5. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. IEEE Proc. **94**(2), 370–382 (2006)
6. Barenghi, A., Breveglieri, L., Koren, I., Naccache, D.: Fault injection attacks on cryptographic devices: theory, practice and countermeasures. Proc. IEEE **100**(11), 3056–3076 (2012)
7. Berzati, A., Canovas, C., Dumas, J.G., Goubin, L.: Fault attacks on RSA public keys: left-to-right implementations are also vulnerable. In: Fischlin, M. (ed.) RSA Cryptographer's Track (CT-RSA 2009). pp. 414–428. LNCS 5473, Springer-Verlag, Berlin, Heidelberg (2009)
8. Biehl, I., Meyer, B., Müller, V.: Differential fault attacks on elliptic curve cryptosystems (extended abstract). In: Bellare, M. (ed.) Advances in Cryptology – CRYPTO 2000. pp. 131–146. LNCS 1880, Springer-Verlag, Berlin, Heidelberg (2000)
9. Blömer, J., Otto, M., Seifert, J.P.: Sign change fault attacks on elliptic curve cryptosystems. In: Fault Diagnosis and Tolerance in Cryptography. pp. 36–52. LNCS 4236, Springer-Verlag, Berlin, Heidelberg (2006). https://doi.org/10.1007/11889700_4
10. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Advances in Cryptology – EUROCRYPT '97. pp. 37–51. LNCS 1233, Springer-Verlag, Berlin, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_4
11. Brier, E., Chevallier-Mames, B., Ciet, M., Clavier, C.: Why one should also secure RSA public key elements. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems (CHES 2006). pp. 324–338. LNCS 4249, Springer-Verlag, Berlin, Heidelberg (2006)
12. Burchard, J., Gay, M., Ekossono, A.M., Horáček, J., Becker, B., Schubert, T., Kreuzer, M., Polian, I.: Autofault: towards automatic construction of algebraic fault attacks. In: 2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 65–72. IEEE, Piscataway, NJ, USA (2017). https://doi.org/10.1109/FDTC.2017.13

13. Certicom Research Group, .: Sec 2: Recommended elliptic curve domain parameters (sep 2000), https://www.secg.org/SEC2-Ver-1.0.pdf
14. Ciet, M., Joye, M.: Elliptic curve cryptosystems in the presence of permanent and transient faults. Des. Codes Cryptogr. **36**(1), 33–43 (Jul 2005). https://doi.org/10.1007/s10623-003-1160-8
15. Dehbaoui, A., Dutertre, J.M., Robisson, B., Tria, A.: Electromagnetic transient faults injection on a hardware and a software implementation of AES. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 7–15. IEEE, Piscataway, NJ, USA (2012)
16. Dottax, E.: Fault attacks on NESSIE signature and identification schemes (2002), www.cosic.esat.kuleuven.be/nessie/nessie/reports/phase2/SideChan_1.pdf, nESSIE Public Report
17. Ebeid, N., Hasan, M.A.: On binary signed digit representations of integers. Des. Codes Cryptogr. **42**(1), 43–65 (Jan 2007). https://doi.org/10.1007/s10623-006-9014-9
18. Eberle, H., Wander, A., Gura, N., Chang-Shantz, S., Gupta, V.: Architectural extensions for elliptic curve cryptography over $\mathrm{gf}(2^m)$ on 8-bit microprocessors. In: 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors. pp. 343–349. ASAP '05, IEEE, Piscataway, NJ, USA (2005). https://doi.org/10.1109/ASAP.2005.15
19. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory **31**(4), 469–472 (1985)
20. Ernst, M., Jung, M., Madlener, F., Huss, S.A., Blümel, R.: A reconfigurable system on chip implementation for elliptic curve cryptography over $\mathrm{gf}(2^n)$. In: Cryptographic Hardware and Embedded Systems – CHES 2002. pp. 381–399. LNCS 2523, Springer-Verlag, London, UK (2003). https://doi.org/10.1007/3-540-36400-5_28
21. Fouque, P.A., Lercier, R., Réal, D., Valette, F.: Fault attack on elliptic curve montgomery ladder implementation. In: Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography. pp. 92–98. FDTC, IEEE, Piscataway, NJ, USA (2008). https://doi.org/10.1109/FDTC.2008.15
22. Gaspar, L.: Cryptoprocessor-architecture, programming and evaluation of the security. Ph.D. thesis, Université Jean Monnet, Saint-Etienne, France (2012)
23. Giraud, C., Knudsen, E.W.: Fault attacks on signature schemes. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) 9th Australasian Conf. on Information Security and Privacy – ACISP 2004. pp. 478–491. LNCS 3108, Springer-Verlag, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_41
24. Giraud, C., Knudsen, E.W., Tunstall, M.: Improved fault analysis of signature schemes. In: Gollmann, D., Lanet, J., Iguchi-Cartigny, J. (eds.) 9th Int. Conf. on Smart Card Research and Advanced Appl. – CARDIS 2010. pp. 164–181. LNCS 6035, Springer-Verlag, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12510-2_12
25. Giraud, C., Thiebeauld, H.: A survey on fault attacks. In: Quisquater, J., Paradinas, P., Deswarte, Y., El Kalam, A. (eds.) Smart Card Research and Advances Applications VI. pp. 159–176. IFIP 153, Springer-Verlag, Boston (2004). https://doi.org/10.1007/1-4020-8147-2_11
26. Hurley, B., Hurley, T.: Group ring cryptography. Int. J. Pure Appl. Math. **69**(1), 67–86 (2011)
27. Joye, M., Quisquater, J.J., Bao, F., H., D.R.: Rsa type signtures in the presence of transient faults. In: Darnell, M. (ed.) 6th IMA Int. Conf. on Cryptography and Coding. pp. 155–160. LNCS 1355, Springer-Verlag, Berlin, Heidelberg (1997)

28. Joye, M., Tunstall, M.: Fault Analysis in Cryptography. Springer-Verlag, Berlin, Heidelberg (2012)
29. Kahrobaei, D., Koupparis, C., Shpilrain, V.: Public key exchange using matrices over group rings. Groups - Complexity - Cryptology **5**(1), 97–115 (2013)
30. Kahrobaei, D., Koupparis, C., Shpilrain, V.: A CCA secure cryptosystem using matrices over group rings. In: Algorithmic Problems of Group Theory, Their Complexity, and Applications to Cryptography. Contemp. Math., vol. 633, pp. 73–80. Amer. Math. Soc., Providence, USA (2015)
31. Kim, C.H., Bulens, P., Petit, C., Quisquater, J.: Fault attacks on public key elements: application to dlp-based schemes. In: Mjølsnes, S., Mauw, S., Katsikas, S. (eds.) 5th European Workshop on Public Key Infrastructure: Theory and Practice – EuroPKI 2008. pp. 182–195. LNCS 5057, Springer-Verlag, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69485-4_13
32. Kim, C.H., Quisquater, J.: Fault attacks for CRT based RSA: new attacks, new results, and new countermeasures. In: Sauveron, D., Markantonakis, C., Bilas, A., Quisquater, J. (eds.) Information Security Theory and Practices (WISTP 2007). pp. 215–228. LNCS 4462, Springer-Verlag, Berlin, Heidelberg (2007)
33. Koblitz, N.: Elliptic curve cryptosystems. Math. Comp. **48**(177), 203–209 (1987)
34. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology – CRYPTO'99. pp. 388–397. LNCS 1666, Springer-Verlag, Berlin, Heidelberg (1999)
35. Koschuch, M., Lechner, J., Weitzer, A., Großschädl, J., Szekely, A., Tillich, S., Wolkerstorfer, J.: Hardware/software co-design of elliptic curve cryptography on an 8051 microcontroller. In: Cryptographic Hardware and Embedded Systems – CHES 2006. pp. 430–444. LNCS 4249, Springer-Verlag, Berlin, Heidelberg (2006). https://doi.org/10.1007/11894063_34, https://iacr.org/archive/ches2006/34/34.pdf
36. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Advances in Cryptology – CRYPTO '97. pp. 249–263. LNCS 1294, Springer-Verlag, Berlin, Heidelberg (1997). https://doi.org/https://doi.org/10.1007/BFb0052240, https://link.springer.com/chapter/10.1007/BFb0052240
37. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks - Revealing the Secrets of Smart Cards. Springer-Verlag, Berlin, Heidelberg (2007)
38. Mohammed, E.: A diffie-hellman key exchange protocol using matrices over noncommutative rings. Groups - Complexity - Cryptology **4**(1), 167–176 (2012)
39. NIST: Digital Signature Standard (DSS), FIPS PUB 186-4 (2013), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
40. Polian, I., Regazzoni, F.: Counteracting malicious faults in cryptographic circuits. In: 22nd IEEE European Test Symp. (ETS). IEEE, Piscataway, NJ, USA (2017). https://doi.org/https://doi.org/10.1109/ETS.2017.7968230
41. Rivain, M.: Fast and regular algorithms for scalar multiplication over elliptic curves. IACR Cryptology ePrint Archive, **Report 2011/338** (2011), https://eprint.iacr.org/2011/338.pdf
42. Simpson, E., Schaumont, P.: Offline hardware/software authentication for reconfigurable platforms. In: Cryptographic Hardware and Embedded Systems – CHES 2006. pp. 311–323. LNCS 4249, Springer-Verlag, Berlin, Heidelberg (2006). https://doi.org/https://doi.org/10.1007/11894063_25, https://link.springer.com/chapter/10.1007/11894063_25

43. van Woudenberg, J.G.J., Witteman, M.F., Menarini, F.: Practical optical fault injection on secure microcontrollers. In: 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 91–99. IEEE, Piscataway, NJ, USA (2011)