Sign Change Fault Attacks On Elliptic Curve Cryptosystems

Johannes Blömer[1], Martin Otto[1,3], Jean-Pierre Seifert[2]

[1]: Paderborn University
Institute for Computer Science
33095 Paderborn, Germany
{bloemer,martinmo}@upb.de

[2]: Intel Corporation
Virtualization & Trust Lab - CTG
2111 NE 25th Avenue, M/S JF2-55
Hillsboro, OR 97124-5961, USA
jean-pierre.seifert@intel.com

Abstract. We present a new type of fault attacks on elliptic curve scalar multiplications: Sign Change Attacks. These attacks exploit different number representations as they are often employed in modern cryptographic applications. Previously, fault attacks on elliptic curves aimed to force a device to output points which are on a cryptographically weak curve. Such attacks can easily be defended against. Our attack produces points which do not leave the curve and are not easily detected. The paper also presents a revised scalar multiplication algorithm that protects against Sign Change Attacks.
Keywords: elliptic curve cryptosystem, fault attacks, smartcards.

1 Introduction

Secure cryptographic applications require a secure platform, which is not offered by today's desktop computers. Consequently, sensitive applications, especially for digital signatures, are deployed on smartcards. Smartcards are tamper-resistant and not threatened by viruses and other malicious code. However, smartcards must adhere to the laws of physics, a fact that can be exploited by an adversary to collect additional information about their computations using side-channel information. The most prominent side-channels are given by timing measurements, power consumption measurements, and faulty outputs.

In 1997, Boneh, DeMillo and Lipton ([BDL01]) introduced fault attacks, which exploit faulty outputs. They showed how to use errors in the computation of an RSA signature to recover the secret key. Today, several different methods to purposely induce faults into devices and memory structures have been reported (e.g., [AK96], [SA02]). As it is a quite natural idea to extend the results to other group based cryptosystems, [BMM00] show how to exploit errors in elliptic curve scalar multiplications. This result has been refined in [CJ03].

All fault attacks on elliptic curve cryptosystems presented so far ([BMM00], [CJ03]) tried to induce faults into the computation of a scalar multiplication

[3] Supported by the DFG graduate school No. 693 and the PaSCo Institute, Paderborn.

kP on the elliptic curve E such that the computation no longer takes place on the original curve E. By changing the base point P or an intermediate point randomly, by changing the curve parameters of E, or by changing the defining field, the operations leave the group defined by the elliptic curve E. Instead the scalar multiplication is done on a different curve $\tilde{E}$ and/or with a different base point $\tilde{P}$. Then the so-called pseudo-addition can be used to recover the secret key if the point $k \cdot \tilde{P}$ on the new curve $\tilde{E}$ allows to solve the discrete logarithm problem at least partially. The disadvantage (or advantage) of the proposed attacks is that there is an obvious and efficient countermeasure: simply check whether the result is a point on the original curve E or not.

In this paper, we present a new type of fault attacks on elliptic curve scalar multiplication, Sign Change Attacks. Our attack does not change the original curve E and works with points on the curve E. We show how sign changes of intermediate points can be used to recover the secret scalar factor. Our attack leads to a faulty output that is a valid point on the original elliptic curve. Then we can use an algorithm similar to the one presented for RSA in [BDL01] to recover the secret scalar factor in expected polynomial time. We present our attack for the NAF-based left-to-right repeated doubling algorithm, because here Sign Change Faults seem to be easier to realize than for other repeated doubling variants (see Section 5). However, we stress the fact that the attack can also be used against other scalar multiplication algorithms, e.g., the right-to-left version, binary expansion based repeated doubling, and the Montgomery ladder ([Mon87]) if the y-coordinate is used.

Our attacks show that the basic ideas of [BDL01] carry over to elliptic curve cryptosystems as well. Clearly, the standard countermeasures described above, namely checking whether the result lies on the original curve, fail to detect Sign Change Attacks. In fact, they even support Sign Change Attacks by holding back a great variety of faulty results if they have been caused by errors other than Sign Change Faults. This allows an adversary to use a less precise attack setting.

We also present a revised version of the basic scalar multiplication algorithm for elliptic curves that is secure against Sign Change Attacks in Section 4. Our countermeasure is motivated by a similar countermeasure by Shamir against attacks on CRT-RSA exponentiations ([Sha99]). We use the original elliptic curve together with a second small curve, which allows to define a larger "combined curve", where the desired scalar multiplication is performed. Using this combined curve, one can check the final result efficiently. We show that this technique secures all standard repeated doubling algorithms against Sign Change Attacks and previously reported attacks. Our analysis proves ad hoc security against these attacks only, it does not provide a general security proof or security reduction. Research on fault attacks has not yet established a mathematical framework to allow general security claims.

One can also use randomization schemes to counteract a differential fault attack with Sign Change Faults. However, smartcard certification authorities often require that algorithms are secure against fault attacks even without random-

ization. Moreover, randomization schemes that only randomize the base point are not guaranteed to counteract an SCA, e.g., Coron's third countermeasure in [Cor99, §5.3] or the proposed elliptic curve isomorphism in [JT01, §4.1]. Alternatively, some scalar multiplication algorithms like the Montgomery ladder ([Mon87]) can be used without the y-coordinate. Therefore, these methods cannot be attacked by a Sign Change Attack. However, patent issues prevent the usage of the Montgomery ladder and endorse the widespread use of variants of the standard repeated doubling algorithm, mostly based on the NAF.

The paper is organized as follows: After briefly recalling the basics of elliptic curve arithmetic, we present the Sign Change Attack on Elliptic Curve Scalar Multiplication in Section 3. Section 4 is devoted to presentation and analysis of the proposed countermeasure. In Section 5, we discuss methods to carry out Sign Change Faults in practice. Section 6 concludes the paper. Since the main contribution of this paper is the presentation of the new Sign Change Faults and the countermeasure presented in Section 4, we concentrate in this extended abstract on Sections 4 and 5.

2 Elliptic Curve Cryptography

An elliptic curve over a field $\mathbb{F}_p$ with $p > 3$ is defined as the set of points $(x : y : z) \in \mathbb{F}_p^3$ that satisfy the projective Weierstraß equation

$$y^2 z \equiv x^3 + Axz^2 + Bz^3 \bmod p. \tag{1}$$

Moreover, $\mathcal{O}$ denotes the point at infinity $(0 : 1 : 0)$. The points of E form an additive group. The elliptic curve E as well as points on E can be expressed in a variety of coordinate representations, e.g., affine coordinates, projective coordinates, Jacobian coordinates, or Hessian coordinates. This paper concentrates on projective representations as defined above. As we do not need to consider the projective addition formula in detail, we refer the reader to standard literature for a description of the actual computation of the sum of two projective points. A nice overview of several addition formulas can be found in [CMO98].

In cryptosystems based on elliptic curves, e.g., the ElGamal cryptosystem and its variants, a crucial computation is the scalar multiplication of a public base point P with a secret scalar factor k. Attacks aim to recover the value k. Several implementations of fast scalar multiplication algorithms have been presented in the literature. In Algorithm 1, we present a left-to-right version of the well known repeated doubling algorithm to present our attack. Algorithm 1 already implements a standard countermeasure against random fault attacks in Line 5. It protects against previously proposed fault attacks on elliptic curve cryptosystems ([BMM00], [CJ03]). For all subsequent considerations, we will always assume that this standard countermeasure is applied.

Algorithm 1 (NAF-based Repeated Doubling on Elliptic Curve E)
Input: *A point P on E, and a secret key $1 < k < ord(P)$ in non-adjacent form, where n denotes the binary length of k, i.e. the number of bits of k*
Output: kP *on* E
1 Set $\mathsf{Q_n} := \mathcal{O}$
2 For i from $\mathsf{n}-1$ downto 0 do
3 Set $\mathsf{Q'_i} := 2\mathsf{Q_{i+1}}$
4 If $\mathsf{k_i} = 1$ then set $\mathsf{Q_i} := \mathsf{Q'_i} + \mathsf{P}$
else if $\mathsf{k_i} = -1$ then set $\mathsf{Q_i} := \mathsf{Q'_i} - \mathsf{P}$
else set $\mathsf{Q_i} := \mathsf{Q'_i}$
5 If $\mathsf{Q_0}$ is not on E then set $\mathsf{Q_0} := \mathcal{O}$
6 **Output** $\mathsf{Q_0}$

In Algorithm 1, we use the non-adjacent form (NAF) representation of the secret scalar k. The performance of most variants of the the classical repeated doubling algorithm will improve if the scalar k is recoded into non-adjacent form (NAF). The 2-NAF uses digits from $\{-1, 0, 1\}$ and ensures that no two adjacent digits are non-zero. It achieves a higher ratio of zeros to non-zeros. For details on the NAF, see [Boo51], or [JY00]. Using the NAF, subtractions are introduced. Since negating a point on an elliptic curve simply means to change the sign of the y-coordinate, subtractions are cheap operations on elliptic curves. The savings using repeated doubling based on the NAF are 11.11% on average (see [MO90]).

3 The Sign Change Attack on Elliptic Curve Repeated Doubling

Previous fault attacks on elliptic curve scalar multiplication used the fact that a pertubated point is not a valid point on the given curve with high probability. However, such a situation can be easily detected and defended against. In the following, we present a new type of faults, Sign Change Faults. They allow to recover the secret scalar factor of a scalar multiplication operation. Section 5 will investigate how such faults can be induced. It will be shown that these attacks are practical.

Our Fault Model. We assume that an adversary is able to induce a *Sign Change Fault* (SCF) on a specific elliptic curve point used in Algorithm 1. A Sign Change Fault changes the sign of the y-coordinate of an attacked point, e.g., Q'_i on E, such that $Q'_i \mapsto -Q'_i$. The adversary does not know in which iteration of the loop the error occurs. However, we assume that the loop iteration determined by i is chosen i.i.d. according to the uniform distribution. Throughout this paper, we denote the correct final result by Q and a faulty final result by $\tilde{Q}$.

Elliptic curves defined over prime fields, which are recommended by current standards such as ANSI, SEC, and IEEE, have prime order or use a subgroup of prime order. Therefore, we will assume this property for our curves as well. It implies that any point $P \neq \mathcal{O}$ on E must have the same (large) prime order. We will use this assumption frequently.

We state our attack using an algorithm similar to the attack presented by Boneh, DeMillo and Lipton in [BDL01] on RSA. Similar to [BDL01], we need a polynomial number of faulty outputs for the same inputs to achieve a sufficiently high success probability. We use the following result from [BDL01] to bound the number of necessary faulty outputs needed by our attack.

Fact 2 (Number of Necessary Attacks) *Let $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ and let M be the set of all contiguous intervals of length $m < n$ in x. If $c = (n/m) \cdot \log(2n)$ bits of x are chosen uniformly independently at random, then the probability that each interval in M contains at least one chosen bit is at least $1/2$.*

3.1 Sign Change Attack on Q'_i in Line 4.

All of the variables in Lines 3 and 4 can be successfully attacked with a Sign Change Attack (SCA). In the following, we present the attack on the variable Q'_i in Line 4 during some loop iteration $0 \leq i \leq n-1$.

The basic idea of our attack algorithm is to recover the bits of k in pieces of $1 \leq r \leq m$ bits. Here, m is chosen to reflect a trade-off between the number of necessary faulty results derived from Fact 2 and the approximate amount 2^m of offline work. Throughout this paper, we assume that $2^m \ll \#E$. To motivate the algorithm, assume that a faulty value $\tilde{Q}$ is given that resulted from an SCF in Q'_i. We have

$$\tilde{Q} = -2^i Q'_i + \sum_{j=0}^{i} k_j \cdot 2^j \cdot P = -Q + 2L_i(k) \quad \text{with } L_i(k) := \sum_{j=0}^{i} k_j 2^j P \qquad (2)$$

On the right hand side of Equation (2), the only unknown part is $L_i(k)$, which defines a multiple of P. If only a small number of the signed bits $k_0, k_1, \ldots, k_i$ used in that sum is unknown, these bits can be guessed and verified using Equation (2). This allows to recover the signed bits of k starting from the LSBs. Moreover, due to the fact that $Q = L_i(k) + H_{i+1}(k)$, where $H_{i+1}(k) := \sum_{j=i+1}^{n-1} k_j 2^j P$, it is also possible to recover the signed bits of k starting from the MSBs. As we assume that errors are induced uniformly at random, an adversary may choose freely between these two recovery strategies. In the following, we will use the LSB version based on Equation (2). We assume that both Q and $\tilde{Q}$ are known. The complete attack is stated as the following algorithm.

Algorithm 3 (The Sign Change Attack on Q'_i)

Input: *Access to Algorithm 1, n the length of the secret key $k > 0$ in non-adjacent form, $Q = kP$ the correct result, m a parameter for acceptable amount of offline work.*

Output: *k with probability at least $1/2$.*

 # Phase 1: Collect Faulty Outputs

1 Set c := (n/m) · log(2n)

2 Create c faulty outputs of Alg. 1 by inducing a SCF in Q_i' for random values of i.
3 Collect the set $S = \{\tilde{Q} \mid \tilde{Q} \neq Q$ is a faulty output of Algorithm 1 on input $P\}$.
Phase 2: Inductive Retrieval of Secret Key Bits
4 Set $s := -1$ indicating the number $s + 1$ of known bits of k.
5 While $(s < n - 1)$ do
Compute the known LSB part.
6 Set $L := 2 \sum_{j=0}^{s} k_j 2^j P$
Try all possible bit patterns with length $r \leq m$.
7 For all lengths $r = 1, 2, \ldots m$ do
8 For all valid NAF-patterns $x = (x_{s+1}, x_{s+2}, \ldots, x_{s+r})$ with $x_{s+r} \neq 0$ do
Compute and verify the test candidate T_x
9 Set $T_x := L + 2 \sum_{j=s+1}^{s+r} x_j 2^j P$
10 for all $\tilde{Q} \in S$ do
11 if $\left(T_x - \tilde{Q}\right) = Q$ then
12 conclude that $k_{s+1} = x_{s+1}, k_{s+2} = x_{s+2}, \ldots, k_{s+r} = x_{s+r}$,
13 set $s := s + r$, and continue at Line 5
Handle a Zero Block Failure
14 If no test candidate satisfies the verification step, then
15 assume that $k_{s+1} = 0$ and set $s := s + 1$.
16 Verify $Q = kP$. If this fails then output "failure".
17 **Output** k

Comment. Algorithm 3 has been abbreviated for clarity in two minor details. On the one hand, the highest iteration that suffered a SCF in Line 2 of Algorithm 3 does not need to be the last iteration $n - 1$. However, since we assume the "lucky case" of Fact 2, this special case can be handled efficiently by an exhaustive search for at most m bits.

Furthermore, it is clear that given n as the length of the NAF of k, Algorithm 3 does not need to test patterns whenever $s + r \geq n$. Note that s indicates that the $s + 1$ least significant bits $k_0, k_1, \ldots, k_s$ are known. In fact, we may assume that the most significant bit of k is $k_{n-1} = 1$, otherwise n cannot be uniquely defined. Therefore, we may assume w.l.o.g. that $s + r < n - 1$. Note that we assume $k > 0$. We also assume that $(k_0, k_1, \ldots, k_s, x_{s+1}, \ldots, x_{s+r})$ is always in valid NAF.

We will prove the success of Algorithm 3 in two lemmas. First, we will show that only a correct guess for the pattern of k can satisfy the verification step in Line 11. Then, we will show that Algorithm 3 will always correctly recover at least the next unknown bit of k. The analysis of these two cases is very similar to the analysis of a similar attack on RSA, presented in [BDL01]. Therefore, we omit the proofs of the two lemmas and the summarizing theorem. They will be presented in detail in the full version. Before stating the results, we introduce Zero Block Failures.

Definition 4 (Zero Block Failure) *Assume that Algorithm 3 already recovered the $s+1$ least significant signed bits $k_0, k_1, \ldots, k_s$ of k. If the signed bits k_{s+1},*

$k_{s+2}, \ldots, k_{s+r}$ are all zero and all Sign Change Faults that happened in iterations $s+1, \ldots, s+m$ really occurred in the first r iterations $s+1, s+2, \ldots, s+r$, the situation is called a Zero Block Failure*.*

A Zero Block Failure is named after the fact that errors in a block of zeros will not be detected as errors within that block. Equation (2) shows that for any s, $L_s(k) = L_{s+1}(k) = \ldots = L_{s+r}(k)$ for all sequences $k_{s+1} = 0, k_{s+2} = 0, \ldots, k_{s+r} = 0$. In this case, the values $\tilde{Q}_1 = -Q + 2L_s(k)$ and $\tilde{Q}_2 = -Q + 2L_{s+r}(k)$ are equal. Therefore, given $\tilde{Q} = -Q + 2L_s(k)$, Algorithm 3 cannot determine how many zero bits — if any — follow k_s. Hence, tailing zeros must be neglected, because their number cannot be determined correctly. This is the reason why Algorithm 3 only tests patterns x which end in ± 1 in Line 8.

In Algorithm 3, we may have one of two cases in each iteration of the loop of Lines 5–15. First, we may encounter a test pattern, which satisfies the verification step in Line 11. Second, no test pattern may satisfy the verification step. The following two lemmas show that Algorithm 3 recovers at least one bit of k correctly in either case.

Lemma 5 (No False Positives) *We assume that the bits $k_0, k_1, \ldots, k_s$ of k have already been computed by Algorithm 3. If Algorithm 3 computes a test bit pattern $x = (x_{s+1}, x_{s+2}, \ldots, x_{s+r})$, $r \leq m$, such that T_x satisfies the verification step in Line 11 for some $\tilde{Q} \in S$, then x is the correct bit pattern, i.e., $x_j = k_j$ for all $s + 1 \leq j \leq s + r$.*

Lemma 6 (Correct Recovery) *We assume that the bits $k_0, k_1, \ldots, k_s$ of k have already been computed by Algorithm 3. Furthermore, we assume that a Sign Change Fault was induced into the intermediate value Q'_i in Line 4 of Algorithm 1 for some $i \in \{s + 1, s + 2, \ldots, s + m\}$.*

Then, in order to recover the next signed bit k_{s+1}, Algorithm 3 will be in one of two cases: In the first case, it finds a test bit pattern $x = (x_{s+1}, x_{s+2}, \ldots, x_{s+r})$, $r \leq m$, that satisfies the verification step in Line 11 and concludes that $k_j = x_j$ for all $s + 1 \leq j \leq s + r$ in Line 12. In the second case, it detects a Zero Block Failure and concludes that $k_{s+1} = 0$ in Line 15. In both cases, the conclusion is correct and between 1 and r bits of k are recovered correctly.

The results of the previous lemmas are summarized in the following theorem. It is a straightforward result from the two previous lemmas, combined with a simple count of operations performed by Algorithm 3.

Theorem 7 (Success of the Proposed Sign Change Attack) *Algorithm 3 succeeds to*
recover the secret scalar multiple k of bit length n in time $O(n \cdot 3^m \cdot c \cdot M)$ with probability at least $1/2$. Here, $c = (n/m) \cdot \log(2n)$ and M is the maximal cost of a full scalar multiplication or a scalar multiplication including the induction of a Sign Change Fault.

The results of Theorem 7 carry over similarly to Sign Change Attacks on all other variables used inside the loop in Algorithm 1. The ideas presented in the attack also apply to the NAF-based right-to-left repeated squaring version and to the binary expansion based versions. Sign Change Attacks can also be used against the Montgomery ladder [Mon87] if the y-coordinate is used.

4 Countermeasures

As explained in the introduction, previously proposed countermeasures cannot be used to defend against Sign Change Faults. Therefore, we propose a modified scalar multiplication algorithm presented as Algorithm 8 as an alternative countermeasure against Sign Change Attacks (SCA). It adds little overhead at the benefit of checking the correctness of the final result. Moreover, it can be based on any scalar multiplication algorithm which does not need field divisions. We will present our countermeasure in the remainder of this section and analyze it using the NAF-based version presented as Algorithm 1. The countermeasure has been motivated by Shamir's countermeasure against attacks on CRT-RSA exponentiations [Sha99].

We first explain the basic idea of the countermeasure. For the modified algorithm, we assume that the curve $E = E_p$ is defined over a prime field $\mathbb{F}_p$, i.e., we have $E_p := E(\mathbb{F}_p)$. Furthermore, we choose a small prime t of about $60 - 80$ bits to form the "small" curve $E_t := E(\mathbb{F}_t)$. E_t does not depend on E_p. Given both curves, we define a "combined" elliptic curve E_{pt} over the ring $\mathbb{Z}_{pt}$. This curve E_{pt} is defined with parameters A_{pt} and B_{pt} such that $A_{pt} \equiv A_p \bmod p$, $A_{pt} \equiv A_t \bmod t$ and $B_{pt} \equiv B_p \bmod p$, $B_{pt} \equiv B_t \bmod t$. Here, A_p and A_t denote the A-parameters and B_p and B_t denote the B-parameters in Equation (1) of E_p and E_t respectively. Both A_{pt} and B_{pt} can be easily computed using the Chinese Remainder Theorem (CRT). We also choose a base point P_t on E_t and use the combined point P_{pt} as the base point for the scalar multiplication in E_{pt}. Here, P_{pt} is computed using the CRT in the same manner as A_{pt} and B_{pt} above. Computing $Q = kP_{pt}$ on E_{pt} allows to verify the result on the small curve E_t.

Algorithm 8 (Sign Change Attack Secure Scalar Multiplication)
Input: *A point P on E_p, and a secret key $1 < k < ord(P)$, where n denotes the binary length of k, i.e., the number of bits of k*
Output: kP *on* E_p
\# **offline initialization (i.e., at production time)**
1 Choose a prime t and an elliptic curve E_t
2 Determine the combined curve E_{pt}
\# **main part**
3 Set Q := kP_{pt} on E_{pt} (e.g., using Algorithm 1)
4 Set R := kP_t on E_t (e.g., using Algorithm 1)
5 If R $\neq$ Q mod t then **output** "failure".
6 Else **output** Q on E_p

Scalar Multiplication is used twice, once in Line 3 and once in Line 4. For the algorithm used, we assume that it features a check of the final result that returns $\mathcal{O}$ if the result is not a valid point on the curve (e.g., Line 5 of Algorithm 1). In the case where E_{pt} is used, we assume for simplicity that this check is performed both modulo p and modulo t, i.e., both on E_p and on E_t.

On the Choice of E_p and E_t. For the security of our countermeasure against Sign Change Attacks, we assume that both E_p and E_t have prime order. Both curves are chosen independently, which allows to use recommended curves (e.g., by [SEC00]) for E_p. The security analysis will show that the security depends on the order of P_t on E_t. This does not require E_t to be secret. Moreover, it also does not require $\#E_t$ to be prime. It is sufficient to choose a curve E_t and a point P_t such that the order of P_t on E_t is large. We will specify a minimal size for the order of P_t on E_t later. Finding such a curve E_t is feasible as shown in [BSS99, §VI.5].

4.1 Analysis of the Countermeasure.

It is easily shown that Algorithm 8 computes the correct result if no error occurs. This is evident from modular arithmetic.

It remains to show that the proposed algorithm is secure against Fault Attacks. We only consider ad-hoc security for our proofs, i.e., we only prove security against known Fault Attacks. Research on fault attacks has not yet established a mathematical framework to allow general security claims. For our analysis, we assume that Algorithm 1 has been chosen as the scalar multiplication algorithm, although the result holds for other scalar multiplication algorithms as well. To defend against previously proposed fault attacks, the standard countermeasure introduced in Section 2 has been included as an integral part of Algorithm 1.Therefore, we concentrate on security against Sign Change Attacks on Line 3 of Algorithm 8 only.

We use the same fault model as in Section 3, i.e., a Sign Change Fault can be induced in any intermediate variable used by the scalar multiplication $Q = kP$ on E_{pt}. Sign Change Faults can only be induced in points of elliptic curves, the scalar k cannot be attacked. Furthermore, we assume that only a single SCF can be induced during each computation of kP. We do not consider multiple correlated attacks, since such attacks are not a realistic scenario. The adversary can target a specific variable, e.g., Q'_i, but he cannot target a specific iteration i. As we are interested to show that a faulty value is returned with negligible probability, we only need to investigate Sign Change Attacks on the computation in Line 3 of Algorithm 8. Attacks in Line 4 cannot yield a faulty output as Q is not changed by Line 4. We first investigate the basic requirement for an error to be undetected by the countermeasure in Line 5.

Lemma 9 (Undetectable Sign Change Faults) *Let $Q = kP_{pt}$ be the correct result of the scalar multiplication in Line 3 of Algorithm 8 and let $\tilde{Q} = Q + \kappa_i \cdot P \neq Q$ be a faulty result from an attack on Line 3. Let $r_t := \#E_t$ be*

the group order of E_t, assumed to be prime. The faulty result $\tilde{Q}$ passes by the detection mechanism in Line 5, iff $r_t \mid \kappa_i$.

Proof. Let R and Q denote the variables used in Algorithm 8. If $r_t \mid \kappa_i$, we have $\kappa_i P = \mathcal{O}$ on E_t. Therefore, the test "$\mathsf{R} \neq \mathsf{Q}$" in Line 5 of Algorithm 8 yields $kP_t = Q + \mathcal{O}$ on E_t. As the correct result Q satisfies $Q = kP_t$ on E_t, this would not trigger a "failure" output and the faulty value $\tilde{Q}$ would be returned. As $\tilde{Q} \neq Q$ on E_{pt} is assumed, we also have $\tilde{Q} \neq Q$ on E_p. This case results in a faulty output.

If $r_t \nmid \kappa_i$, we must show that $kP_t \neq \tilde{Q}$ on E_t. We know that for the correct value Q, it holds that $\mathsf{R} = Q$ on E_t. If $r_t \nmid \kappa_i$, we have $\kappa_i P \neq \mathcal{O}$ on E_t because r_t is the prime group order. Therefore, the order of P is r_t as well. Consequently, we have $\mathsf{R} \neq \mathsf{Q} = \tilde{Q}$ on E_t and the security alert in Line 5 is triggered. □

Lemma 10 (Number of Undetectable Sign Change Faults) *Let r_t be the group order of E_t, assumed to be prime. Let m be the blocksize used in Algorithm 3. Then a Sign Change Attack on Algorithm 8 needs a blocksize $m \geq \lfloor \log(r_t) \rfloor$ to be successful. Moreover, at most $(n-1)/\lfloor \log(r_t) \rfloor$ many undetectable faulty outputs exist.*

Proof. Assume that a Sign Change Fault was induced into Q'_i for some i, resulting in a faulty output $\tilde{Q}_1$. By Equation (2), we have

$$\tilde{Q}_1 = -Q + 2L_i(k) = Q + \kappa_i P \qquad \text{where} \qquad \kappa_i := -2^{i+2} \sum_{j=i+1}^{n-1} k_j 2^{j-i-1}.$$

We further assume that $r_t \mid \kappa_i$, i.e., $\tilde{Q}_1$ has not been detected as a faulty value according to Lemma 9. We now consider another faulty output $\tilde{Q}_2 \neq \tilde{Q}_1$ collected by Algorithm 3. Let $u \neq i$ denote the fault position, i.e., $\tilde{Q}_2 = Q + \kappa_u P$.

We claim that for all u with $|u - i| \leq \lfloor \log(r_t) \rfloor$, $u - i \neq 0$, it holds that $r_t \nmid \kappa_u$. We consider the two cases $u < i$ and $u > i$. For $u < i$, we have

$$\kappa_u = -2^{u+2} \sum_{j=u+1}^{n-1} k_j 2^{j-u-1} = \kappa_i - 2^{u+2} \cdot \sigma_u, \quad \text{where } \sigma_u := \sum_{j=u+1}^{i} k_j 2^{j-u-1}, \tag{3}$$

and for $u > i$, we have

$$\kappa_u = -2^{u+2} \sum_{j=u+1}^{n-1} k_j 2^{j-u-1} = \kappa_i + 2^{i+2} \cdot \rho_u, \text{ where } \rho_u := \sum_{j=i+1}^{u} k_j 2^{j-i-1}. \tag{4}$$

The value $\tilde{Q}_2$ is only output if it is an undetectable fault that bypassed Line 5 of Algorithm 8. According to Lemma 9, this requires that $r_t \mid \kappa_u$. As we assume that $r_t \mid \kappa_i$, we need to analyze the case that $r_t \mid \sigma_u$ and $r_t \mid \rho_u$ respectively. We first investigate σ_u. Here, we have two cases: Either $\sigma_u = 0$ or $\sigma_u > 0$ over

the integers. If $\sigma_u = 0$ over the integers, we have $\kappa_u = \kappa_i$ and $\tilde{Q}_1 = \tilde{Q}_2$. As this contradicts our assumption that $\tilde{Q}_1 \neq \tilde{Q}_2$, we may assume that σ_u is not equal to 0 over the integers. If the sum in Equation (3) is not equal to 0 over the integers, its absolute value must be at least as large as r_t in order to be a multiple of r_t. The same considerations hold for $u > i$.

Algorithm 3 recovers bits in blocks of at most m bits. If it has found a valid test pattern, it starts at the position immediately following that test pattern and tries to recover the next block of length m starting at this position. If $m < \lfloor \log(r_t) \rfloor$, the arguments above shows that in this block there cannot be a faulty output $\tilde{Q}$ in the list of collected faulty outputs that satisfies the verification step. Therefore, Algorithm 3 needs a minimal blocksize of $m = \lfloor \log(r_t) \rfloor$ in order to be able to reconstruct another faulty output $\tilde{Q}$.

As the fault positions of two undetected faulty outputs $\tilde{Q}_1$ and $\tilde{Q}_2$ are at least $\lfloor \log(r_t) \rfloor$ bits away from each other, we have a maximum of $(n-1)/\lfloor \log(r_t) \rfloor$ many different faulty outputs in the set collected by Algorithm 3. □

Lemma 10 shows that the proposed algorithm secures the scalar multiplication algorithm against the new Sign Change Faults if the group order of $\#E_t$ is large enough. A group order of $\#E_t > 2^{80}$ guarantees that the required block size of $m > 80$ exceeds the acceptable amount of offline work significantly. For many practical applications, $\#E_t > 2^{60}$ should already be enough.

The computational overhead is acceptable. Line 3 requires computations with 30 – 40 % larger moduli (for $l(p) = 192$ and $l(t) = 60$ – 80), Line 4 requires a scalar multiplication on a considerably smaller curve with the scalar factor $k \bmod \#E_t$, which is considerably smaller than k.

As the computations in Line 3 prohibit the use of inversions, projective coordinates must be used. We have stated our results for the basic version of projective coordinates but other weighted projective representations such as Jacobian or Hessian representations will do just as well.

5 Realization of Sign Change Attacks

At first sight, a Sign Change Attack does not seem to be easily performed in a general setting. A random change of the y-coordinate cannot hope to yield $-y$ with non-negligible probability. However, there exist several special yet common settings, where Sign Change Faults can be realized. We will give examples for attacks on NAF-based variants of the scalar multiplication algorithm as well as examples for attacks on certain properties of the crypto co-processor. The latter attacks can be applied to any variant of repeated doubling.

One special way to attack the NAF is offered by the fact that any NAF-based algorithm has to incorporate a conditional branch, where for secret key bit $k_i = 1$ an addition is performed and for secret key bit $k_i = -1$, a subtraction is performed. Currently, a whole zoo of physical attacks is available that targets such conditional decisions, e.g., power spikes or clock glitches ([BCN+04], [ABF+02]). These attacks aim at forcing the conditional statement to choose

the wrong branch. In our case, choosing the wrong branch means to add $-P$ instead of P or vice versa. Although this attack cannot be applied to mount a Sign Change Attack on other intermediate variables, such as Q'_i as analyzed in Section 3, it is an instructive example. Moreover, a Sign Change Attack on P can also be used to recover the secret key, similar to the attack described in Section 3. This attack is also valid for more sophisticated NAF-based repeated doubling variants, which aim to secure the basic scheme against power and timing attacks, e.g., by using dummy operations.

To achieve sign changes of any intermediate variable, consider the following scenario. Many real-world embedded crypto co-processors supporting modular arithmetic most often also rely on the non-adjacent form to speed up the time-critical modular *multiplication* operation, cf. [HP98], [Kor93]. Here, the factors used during the computation of $P_1 + P_2$ for two elliptic curve points P_1 and P_2 are attacked. Any efficient implementation of the NAF must provide special hardware to handle negative numbers, i.e., being able to compute the two's complement of any register used as a multiplicand without time delay, cf. [Sed87], [WQ90], or [Mon85]. This task is trivial to solve by simply inverting every bit sent to the long integer arithmetic unit (ALU) and additionally adding $+1$. Given this functionality, it can be used for an attack.

As a concrete example, we consider such a crypto co-processor, cf. [Sed87], adding simultaneously at least three different operands with a possible sign change in one single instruction. Changing the value of an operand to its negative, i.e., to its two's complement, one usually needs to change only one single bit among the control signals of the corresponding ALU. This is due to the fact that the ALU of most crypto co-processors is, as already explained above, designed to handle automatically the two's complement of any operand. Here, a fault attack can be mounted that results in an SCF.

For concreteness, let us consider the following projective addition formula, cf. [IEE98], for points $P_0 = (X_0 : Y_0 : Z_0)$, $P_1 = (X_1 : Y_1 : Z_1)$:

$$
\begin{array}{llll}
U_0 := X_0 Z_1^2, & S_0 := Y_0 Z_1^3, & U_1 := X_1 Z_0^2, & S_1 := Y_1 Z_0^3, \\
W := U_0 - U_1, & R := S_0 - S_1, & T := U_0 + U_1, & M := S_0 + S_1, \\
Z_2 := W Z_0 Z_1, & X_2 := R^2 - TW^2, & V := TW^2 - 2X_2, & 2Y_2 := VR - MW^3.
\end{array}
$$

Here it becomes clear, that lots of load/store or exchange instructions are needed to realize this formulas involving the original points P_0 and P_1. For example, an implementation could use Y_0 or Y_1 via previous load/store or exchange instructions as a multiplicand in the modular multiplications to compute S_0, or S_1. The attack on Q'_i described in Section 3 can be realized by attacking Y_0 during the computation of S_0. During this preparing load/store or exchange instruction, the corresponding value must go through the ALU. While executing this operation, the handled value is susceptible to an SCF as only a single bit among the control signals must be changed to load/store or exchange the value in its target multiplicand register to $-Y_0$ or $-Y_1$. This yields an SCF by changing one single control signal. Note that [BCN+04] actually describes how to implement

such attacks in practice. A similar consideration also applies to the projective doubling formula.

6 Conclusions and Open Problems

Fault attacks are a significant threat to secure communication based on mobile devices. We have introduced a new type of fault attacks on elliptic curve cryptosystems, Sign Change Attack, which allow attacks with a high success probability, especially for NAF-based repeated doubling algorithms. Current and future cryptosystems based on elliptic curves must be guarded against this type of attacks carefully. As a first step in this direction, a new secure algorithm is presented that withstands Sign Change Attacks with acceptable computational overhead. Attack and countermeasure have been presented in the context of projective coordinates and elliptic curves defined over prime fields. However, both attack and countermeasure also apply to other commonly used representations and defining fields.

Interestingly, the Sign Change Attack presented in this paper does not apply to elliptic curves of characteristic 2. It is an open problem to extend our attack to elliptic curves of characteristic 2.

The results from Section 5 show that the most efficient solutions often pay their performance advantage with security, just like in the case of CRT-RSA [BDL01]. Since Montgomery's version is secure, our attack strengthens the claim from [JY03], that the "Montgomery ladder may be a first-class substitute of the celebrated square-and-multiply algorithm". It is an open problem whether it is possible to successfully attack the Montgomery method where the y-coordinate is not used in a way such that faulty results are created which are valid points on the curve.

References

[ABF+02] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert, *Fault attacks on RSA with CRT: Concrete results and practical countermeasures*, CHES 2002, LNCS, vol. 2523, Springer-Verlag, 2002, pp. 260–275.

[AK96] R. J. Anderson and M. G. Kuhn, *Tamper resistance — a cautionary note*, Proceedings of the Second USENIX Workshop on Electronic Commerce, USENIX Association, 1996, pp. 1 – 11.

[BCN+04] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, *The sorcerer's apprentice guide to fault attacks*, Cryptology ePrint Archive, 2004/100, 2004, `http://eprint.iacr.org/2004/100.pdf`.

[BDL01] D. Boneh, R. A. DeMillo, and R. J. Lipton, *On the importance of eliminating errors in cryptographic computations*, J. Cryptology **14** (2001), no. 2, 101–119.

[BMM00] I. Biehl, B. Meyer, and V. Müller, *Differential fault attacks on elliptic curve cryptosystems*, CRYPTO 2000, LNCS, vol. 1880, Springer-Verlag, 2000, pp. 131–146.

[Boo51] A. D. Booth, *A signed binary multiplication technique*, Quart. Journ. Mech. and Applied Math. **IV** (1951), no. 2, 236–240.

[BSS99] I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*, London Mathematical Society Lecture Note Series, vol. 265, Cambridge University Press, 1999.

[CJ03] M. Ciet and M. Joye, *Elliptic curve cryptosystems in the presence of permanent and transient faults*, Cryptology ePrint Archive, 2003/028, 2003, `http://eprint.iacr.org/2003/028.pdf`.

[CMO98] H. Cohen, A. Miyaji, and T. Ono, *Efficient elliptic curve exponentiation using mixed coordinates*, ASIACRYPT'98, LNCS, vol. 1514, Springer-Verlag, 1998, pp. 51–65.

[Cor99] J.-S. Coron, *Resistance against differential power analysis for elliptic curve cryptosystems*, CHES'99, LNCS, vol. 1717, Springer-Verlag, 1999, pp. 292–302.

[HP98] H. Handschuh and P. Pailler, *Smart card crypto-coprocessors for public-key cryptography*, Proc. of CARDIS '98, LNCS, vol. 1820, Springer-Verlag, 1998, pp. 372–379.

[IEE98] IEEE P1363/D3 (Draft Version 3), *Standard specifications for public key cryptography*, May 1998.

[JT01] M. Joye and C. Tymen, *Protections against differential analysis for elliptic curve cryptography — an algebraic approach*, CRYPTO 2001, LNCS, vol. 2162, Springer-Verlag, 2001, pp. 377–390.

[JY00] M. Joye and S. M. Yen, *Optimal left-to-right binary signed-digit recoding*, IEEE Trans. on Computers **49** (2000), no. 7, 740–748.

[JY03] M. Joye and S.-M. Yen, *The montgomery powering ladder*, CHES 2002, LNCS, vol. 2523, Springer-Verlag, 2003, pp. 291–302.

[Kor93] I. Koren, *Computer arithmetic algorithms*, Prentice-Hall, 1993.

[MO90] F. Morain and J. Olivos, *Speeding up the computations on an elliptic curve using addition-subtractions chains*, Theoretical Informatics and Applications (1990), no. 24, 531–543.

[Mon85] P. L. Montgomery, *Modular multiplication without trial division*, Math. Comp. (1985), no. 44, 519–521.

[Mon87] P. L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Mathematics of Computation **48** (1987), no. 177, 243–264.

[SA02] S. Skorobogatov and R. Anderson, *Optical fault induction attacks*, CHES 2002, LNCS, vol. 2523, Springer-Verlag, 2002, pp. 2–12.

[SEC00] Standards for Efficient Cryptography Group (SECG), *SEC 2: Recommended elliptic curve domain parameters*, 2000, `http://www.secg.org/collateral/sec2_final.pdf`.

[Sed87] H. Sedlak, *The RSA cryptography processor*, EUROCRYPT'87, LNCS, vol. 304, Springer-Verlag, 1987, pp. 95–108.

[Sha99] A. Shamir, *Method and apparatus for protecting public key schemes from timing and fault attacks*, 1999, US Patent No. 5,991,415, Nov. 23, 1999.

[WQ90] D. de Waleffe and J.-J. Quisquater, *CORSAIR, a smart card for public-key cryptosystems*, CRYPTO '90, LNCS, vol. 537, Springer-Verlag, 1990, pp. 503–513.