Sign Change Fault Attacks On Elliptic Curve Cryptosystems

(extended version)

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 provably protects against Sign Change Attacks.

Keywords: elliptic curve cryptosystem, fault attacks, smartcards.

1 Introduction

In 1997, Boneh, DeMillo and Lipton ([BDL01]) reported a new type of side channel attack: fault attacks. 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], [QS02]). As it is a quite natural idea to extend the results to other group based cryptosystems, Biel, Meyer and Müller showed in [BMM00] how to exploit errors in elliptic curve scalar multiplications. This result has been refined by Ciet and Joye 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 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

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

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. Transient faults are easily detected and a faulty output is prevented. Permanent faults can be detected if the curve parameters in memory are stored with redundant information that allows to verify data integrity, e.g., CRC sums.

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). The non-adjacent form (NAF) is a unique signed digit representation of an integer using the digits $\{-1, 0, 1\}$, such that no two adjacent digits are both non-zero. 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 Montgomery's binary method ([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 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 or by imprecise Sign Change Faults. This allows an adversary to use a less precise attack setting for Sign Change Attacks (see Section 5).

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 new algorithm is secure against Sign Change Attacks and previously reported attacks. Our analysis proves 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 randomization. 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 [JT01b, §4.1].

Alternatively, some scalar multiplication algorithms like Montgomery's Binary Method ([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 Montgomery's Binary Method and endorse the widespread use of variants of the standard repeated doubling algorithm, mostly based on the NAF. All these standard repeated doubling algorithms are secured by our countermeasure.

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.

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 $(0 : 1 : 0)$. For coordinates of a point P on a given curve E we write $P[x]$ to denote the x-coordinate and so forth. 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, Hessian coordinates or mixed 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 the 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. This will be further explained in the next section.

Algorithm 1 (NAF-based Repeated Doubling on an 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*

```
# init
1 Set Q := O
# main
2 For i from n − 1 downto 0 do
3     Set Q := 2Q
4     If k_i = 1 then set Q := Q + P else if k_i = −1 then set Q := Q − P
5 If Q is not on E then set Q := O
6 Output Q
```

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, which reduces the number of additions/subtractions in the resulting double-and-add-or-subtract method. For details on the NAF, see [Boo51], [Rei60], [JY00], or [OT04]. 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]).

As shown in the full version of this paper, our attack also applies to other scalar multiplication algorithms, e.g., the right-to-left repeated doubling version or Montgomery's binary method where the y-coordinate is used.

3 The Sign Change Attack on Elliptic Curve Repeated Doubling

Previous fault attacks on elliptic curve cryptosystems usually assume that the attack changes the elliptic curve E to some different curve $\tilde{E}$ ([BMM00], [CJ03]). This happens if an attack changes the coordinates of a point P to some other value $\tilde{P}$. The chance that the resulting point $\tilde{P}$ is a valid point on the same curve E is very low (depending on the fault model). However, scalar multiplication with a faulty base point $\tilde{P}$ can be interpreted as an operation on a different curve $\tilde{E}$. The different curve is defined by the same finite field and the original curve parameter A, but with a different parameter $\tilde{B}$. This is possible, because the curve parameter B from the Weierstraß equation (1) is not used in the addition formula. $\tilde{B}$ can easily be computed using the faulty point $\tilde{P}$.

These attacks succeed in recovering the secret scalar factor k of $k\tilde{P}$ if the discrete logarithm problem is solvable for $k\tilde{P}$ on $\tilde{E}$. However, very simple countermeasures defend against these attacks. First, the curve parameters are validated to counteract permanent faults in memory. Then, the result of the scalar multiplication is checked to be a valid point on the curve. If not, an error output results ([CJ03]). The same attacks apply to faults in the curve parameter A or the defining field $\mathbb{F}_p$. To protect against these attacks, we incorporated the proposed countermeasure in Line 5 of Algorithm 1.

In this section, we present a fault attack that uses a faulty point on the original curve, which cannot be detected by the aforementioned countermeasures. The basic idea of our attack is to induce a fault into a point such that the sign of the point changes, i.e., such that the sign of the y-coordinate of a point on E is flipped. Below, we will define the detailed fault model. Section 5 will investigate how such faults can be induced. It will be shown that these attacks are practical.

Starting from Algorithm 1, we will use indices to distinguish between the variable values in the main loop. We will denote the correct final result by Q and a faulty final result by $\tilde{Q}$. To avoid ambiguity when referring to intermediate values, we rewrite Lines 3 and 4 as

3 Set $Q'_i := 2Q_{i+1}$
4 If $k_i = 1$ then set $Q_i := Q'_i + P$
  else if $k_i = -1$ then set $Q_i := Q'_i - P$
  else set $Q_i := Q'_i$

Our Fault Model. In 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. Later in Section 5, we will discuss methods to induce such Sign Change Faults in practice.

Most elliptic curves defined over prime fields, which are recommended by current standards such as ANSI, IEEE, and SEC [IT03], have prime order, i.e., they contain a prime number of points. Therefore, we will assume this property for our curves as well. This fact 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 to be able to mount $c = (n/m)\log(2n)$ attacks on the same input (P, k, E) to recover k with probability at least $1/2$. Here, m is a parameter that will be defined later. Differently from the attack in [BDL01], we need a correct result Q to verify our test cases — which is indeed a plausible assumption. 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 $0 \leq i \leq n-1$. Afterwards, we will briefly describe how to modify the attack for the other variables.

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

$$\tilde{Q} = -2^i Q'_i + \sum_{j=0}^{i} k_j \cdot 2^j \cdot P = -2^i \sum_{j=i+1}^{n-1} k_j 2^{j-i} P + \sum_{j=0}^{i} k_j \cdot 2^j \cdot P$$

$$
\begin{aligned}
&= -Q + 2L_i(k) \qquad \text{with } L_i(k) := \sum_{j=0}^{i} k_j 2^j P && (2)\\
&= \ Q - 2H_{i+1}(k) \qquad \text{with } H_{i+1}(k) := \sum_{j=i+1}^{n-1} k_j 2^j P = Q - L_i(k). && (3)
\end{aligned}
$$

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, based on Equation (3) 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 $c := (n/m) \cdot \log(2n)$
2 Create c faulty outputs of Algorithm 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 the test candidate* T_x
9   Set $T_x := L + 2\sum_{j=s+1}^{s+r} x_j 2^j P$
\# *Verification Step: Verify the test candidate using Equation* (2)
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 needs not be the last iteration $n-1$. Let j be the maximal i such that the value Q'_i has been attacked. Algorithm 3 only recovers bits of k up to iteration j. Following Fact 2, we assume a "lucky case", which means that we assume that every interval of length m was targeted by at least one SCF. This only guarantees that $j \geq n-m$. Hence, the $m-1$ most significant bits may not be recovered. However, up to $m-1$ missing bits can easily be recovered by exhaustive search if m is small enough. If this fails, the adversary comes to the same conclusion as suggested in Line 16: the guessed bits must be wrong and the attack has failed.

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 second result is based on the assumption that each contiguous interval of m bits was targeted by at least one Sign Change Fault. This represents the "lucky case" of Fact 2. 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}, ..., 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. However, the fact that tailing zeros do not change the value of $\tilde{Q}$ allows us to write $\tilde{Q} = -Q + 2L_i(k)$ with $k_i \neq 0$ for some unknown i if $\tilde{Q} \neq -Q$. Moreover, if it is known that $\tilde{Q} = -Q + 2L_s(k)$, we may specify i in greater detail. In fact, we have $\tilde{Q} = -Q + 2L_i(k)$ with $i := \max\{j \mid k_j \neq 0 \wedge j \leq s\}$. The case where $\tilde{Q} = -Q$, i.e., when the maximum does not exist, can be represented by choosing $i = -1$. This simplification will be used in the following two lemmas.

First, we will investigate the case where a test pattern x satisfies the verification step. This allows Algorithm 3 to recover the next r bits of k.

Lemma 5 (No False Positives) *If Algorithm 3 computes a test bit pattern x such that T_x satisfies the verification step in Line 11 for some $\tilde{Q} \in S$, then x is the correct bit pattern.*

Proof. Assume that the verification step is satisfied for a given test bit pattern $x = (x_{s+1}, \ldots, x_{s+r})$ with $1 \leq r \leq m$ and $x_{s+r} \neq 0$, x in non-adjacent form. We assume that we have a false positive, i.e., the pattern x is different from the corresponding pattern of k, namely $k_{s+1}, k_{s+2}, \ldots, k_{s+r}$. Hence, there must be a faulty result $\tilde{Q} \in S$ that satisfies the verification step in Line 11 together with this x. The verification step in Line 11 yields $\mathcal{O} = T_x - \tilde{Q} - Q$. We use Line 9 of Algorithm 3 to express T_x and Equation (2) to express $\tilde{Q}$ in detail. As explained above, we may assume that $\tilde{Q} = -Q + 2L_i(k)$ with $k_i \neq 0$. We have

$$\mathcal{O} = \left(2 \cdot \sum_{j=0}^{s} k_j 2^j P + 2 \cdot \sum_{j=s+1}^{s+r} x_j 2^j P\right) - \left(-Q + 2 \cdot \sum_{j=0}^{i} k_j 2^j P\right) - Q$$

$$= 2 \cdot \left(\underbrace{\sum_{j=0}^{s} k_j 2^j + \sum_{j=s+1}^{s+r} x_j 2^j}_{R_+} - \underbrace{\sum_{j=0}^{i} k_j 2^j}_{R_-} \right) \cdot P \quad = \quad R_x \cdot P \tag{4}$$

$$\text{where} \quad R_x = 2 \cdot \sum_{j=0}^{\max(i,s+r)} y_j 2^j \quad \text{and} \quad y_j = \begin{cases} 0 & \text{if } j \leq \min(i,s) \\ k_j & \text{if } i < j \leq s \\ (x_j - k_j) & \text{if } s < j \leq \min(i, s+r) \\ x_j & \text{if } \max(i,s) < j \leq s+r \\ -k_j & \text{if } s+r < j \leq i. \end{cases}$$

Equation (4) implies that either $R_x = 0$ or R_x is a multiple of the order of P.
Case 1. Assume that $R_x = 0$. This is easily shown to be impossible. It implies that $R_+ = R_-$, i.e., both sums are valid NAF representations of the same number. As the NAF is unique, this implies that both representations are equal. Hence, all digits are equal in contradiction to the assumption that there is as least one $x_j \neq k_j$ with $s+1 \leq j \leq s+r$.
Case 2. Assume that $R_x \neq 0$ is a multiple of the order of P on E. We know that $\mathrm{ord}(P) = \#E \gg 2^m$ as $\#E$ is prime. Therefore, $\#E$ divides R_x. If $i = -1$, i.e., $\tilde{Q} = -Q$, we have $R_x \cdot P = T_x$. As we may assume that $s + r < n - 1$ as explained above, we have $R_x < \#E$. This contradicts our assumption that $\#E$ divides R_x. If $0 \leq i \leq s$, we have $R_x = 2^{i+2} \cdot R'_x$ with $|R'_x| < 2^{s+r-i} < 2^{n-1-i} \leq k < \#E$. Therefore, $\#E$ cannot divide R_x. If $s + 1 \leq i \leq s + r$, we have $R_x = 2^{s+2} \cdot R'_x$ with $|R'_x| < 2^{m+1}$. Again, $\#E$ cannot divide R_x. For $s + r < i < n - 1$, we have $R_x = 2^{s+2} \cdot R'_x$ with $|R'_x| < 2^{i-s+1} \leq 2^{n-2-s+1} \leq 2^{n-1} \leq k < \#E$. Therefore, $\#E$ cannot divide R_x. The last case, $i = n - 1$, is impossible. It would imply that $Q'_i = \mathcal{O}$ has been attacked, where no Sign Change Fault can be induced, i.e., $\tilde{Q} = Q$. However, we explicitely prevent values $\tilde{Q} = Q$ from being members of the set S in Line 3 of Algorithm 3. Therefore, Case 2 is impossible. □

Lemma 4 has shown that if Algorithm 3 has found a test pattern x, it correctly represents the corresponding bit pattern of k. This result can be used to show that whether Algorithm 3 find a test pattern x or not, it always recovers the correct next bits of k.

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.

Proof. We will investigate the two cases of the lemma separately.
Case 1. Assume that Algorithm 3 finds a test bit pattern $x = (x_{s+1}, x_{s+2}, \ldots, x_{s+r})$ that satisfies the verification step in Line 11. According to Lemma 5, there cannot be a false positive and x correctly represents the bit pattern of k. Therefore, the conclusion $k_j = x_j$ for all $s+1 \leq j \leq s+r$ is correct.
Case 2. Assume that Algorithm 3 does not find a test bit pattern x that satisfies the verification step. In this case, a Zero Block Failure is conjectured by Algorithm 3 and it sets $k_{s+1} = 0$.

We assume that this conjecture is wrong. We know by the assumption in the lemma that at least one of the iterations $s+1, s+2, \ldots, s+m$ was targeted by a Sign Change Fault. Let $\tilde{Q} \in S$ be the faulty output of such an attack, i.e., $\tilde{Q} = -Q + 2L_i(k)$ with $s+1 \leq i \leq s+m$ according to Equation (2). If the conjecture that we have a Zero Block Failure is wrong, we know by Definition 4 that we may choose $\tilde{Q}$ such that at least one of the bits $k_{s+1}, k_{s+2}, \ldots, k_i$ is not zero. This implies that we may write $\tilde{Q} = -Q + 2L_w(k)$ with $w := \max\{j \mid k_j \neq 0 \wedge s+1 \leq j \leq i\}$ as explained above. Now it is easy to see that the test bit pattern $0 \neq x = (k_{s+1}, k_{s+2}, \ldots, k_{w-1}, k_w)$ of length $1 \leq r \leq m$ satisfies the verification step. This means that the value T_x defined in Line 9 of Algorithm 3 correctly represents $2L_w(k)$. Therefore, a valid test pattern x exists and Algorithm 3 will find a value for k_{s+1} in Line 12. Therefore, the assumption that a Zero Block Failure is detected incorrectly must be wrong. □

The results of the previous lemmas are summarized in the following theorem.

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.*

Proof. The results of Lemma 5 and Lemma 6 rely on the assumption that every contiguous interval of length m was targeted by at least one Sign Change Fault in Line 2 of Algorithm 3. According to Fact 2, this assumption holds with probability $1/2$ if $c = (n/m) \cdot \log(2n)$ faulty results are collected. This requires c scalar multiplications with the ability to induce a SCF.

According to Lemma 6, every iteration of the while loop of Algorithm 3 recovers at least a single bit of the secret scalar k. Therefore, at most n iterations are needed. The worst case occurs when $k = 2^{n-1}$ and no SCF was induced into Q_{n-1} while Algorithm 3 created the set of faulty outputs in Line 2. As all bits but the most significant are zero, only Zero Block Failures would occur, allowing to recover only a single bit in each iteration of Algorithm 3.

In a single iteration, Algorithm 3 tests clearly less than 3^m test bit patterns, since every pattern consist of the three digits -1, 0, and 1. Every test bit pattern x yields a test candidate T_x using one scalar multiplication in Line 9. Obviously, some speedups could be applied, e.g., storing T_x allows to compute a new T_x using a single addition. Note that the precomputation of L in Line 6 already represents a speed up. For each test candidate T_x, at most c point additions and comparisons need to be done in Line 11. To present a short result, we simply treat the addition and comparison cost of Line 11 as a full scalar multiplication. If all possible bits of k have been recovered, a last full scalar multiplication must be applied to differentiate between Zero Block Failures and a real failure.

Altogether, the worst case running time of Algorithm 3 is $O((c+n3^m c+1) \cdot M)$ where $c = (n/m) \cdot \log(2n)$ and M is the maximal cost of a full scalar multiplication or a scalar multiplication including the induction of a Sign Change Fault. □

3.2 Other Attacks

The basic idea of Algorithm 3 can also be used for Sign Change Attacks on all other variables used inside the loop in Algorithm 1. An attack on Q_{i+1} in Line 3 yields the same algorithm as an attack on Q'_i in Line 3 or 4. An attack on Q_i in Line 4 yields $\tilde{Q} = 2L_{i-1}(k) - Q$, therefore, Algorithm 3 can also be used for this case.

An attack on the variable Q_{i+1} in Line 3 of Algorithm 1 can be used for another SCA if it is possible to attack just one of the two copies of Q_{i+1}. In this case, an SCF induced into Q_{i+1} yields $Q'_i = \mathcal{O}$ and $\tilde{Q} = L_i(k)$. Algorithm 3 can be used to recover the bits of k starting from the LSBs with a modified verification step $\tilde{Q} - T_x = \mathcal{O}$. This attack does not need to know a correct result Q.

Another attack can be mounted if it is possible to change the sign of the base point P in Line 4. In this case, we have $\tilde{Q} = Q - 2k_i 2^i P$ if the error is transient. Algorithm 3 can be used if the verification step is modified accordingly. Moreover, a larger number of faulty values must be collected as these SCFs require a block length of $m = 1$, which increases the cost significantly. If the error in P is permanent, we have $\tilde{Q} = Q - 2L_i(k)$. Algorithm 3 can also be used for this case. However, attacks on P should be considered much less probable than attacks on the other variables. As P is the base point, it is stored in long term memory. Permanent errors could therefore easily be detected by rechecking.

As shown in the full paper, the ideas presented in this section carry over 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 Montgomery's binary method [Mon87] if the y-coordinate is used.

4 Countermeasures

Clearly, the countermeasure against the attacks proposed in [BMM00] and [CJ03], namely a test if the final result is on the original curve, does not counteract our attacks. The faulty values are bound to be on the curve. Checking whether a point lies on a given curve even helps an attacker to detect and eliminate unsuccessful Sign Change Faults. A quite natural countermeasure against Sign Change Faults, which change the sign of the y-coordinate of an intermediate point, is to use Montgomery's binary method, where the y-coordinate is not needed, cf. [Mon87]. However, several patents prevent its widespread application and endorse the use of other variants of repeated doubling. Additionally, some variants of the ElGamal cryptosystem embed messages in both coordinates (cf. [GG99, § 20.6]) and may wish to use a scalar multiplication method, where both coordinates are computed simultanously. As a consequence, variants of the well-known repeated doubling algorithm, mostly based on the non-adjacent form of the secret scalar factor, are widely employed in modern elliptic curve based systems.

Additionally, as explained in the introduction, randomization could be used to protect against Sign Change Attacks. However, a variety of randomization strategies do not thwart our attack. As an alternative countermeasure against Sign Change Attacks (SCA), we propose a modified scalar multiplication algorithm presented as Algorithm 8. 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 $E_t := E(\mathbb{F}_t)$. We now want to compute the scalar multiple kP in a way such that the result can easily be checked on E_t but also yields the correct result on E_p. To do so, we define an 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, although B_{pt} is not needed in the addition formula. As the base point $P_p := P$ is not guaranteed to exist on E_t, 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 Chinese Remainder Theorem in the same manner as A_{pt} and B_{pt} above, i.e. $P_{pt}[u] \equiv P_p[u] \bmod p$ and $P_{pt}[u] \equiv P_t[u] \bmod t$ for all $u \in \{x, y, z\}$. We will refer to E_{pt} as the "combined curve" of E_p and E_t. 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 $\mathsf{E_t}$

2 Determine the combined curve $\mathsf{E_{pt}}$

\# **main part**

3 Set $\mathsf{Q := kP_{pt}}$ on $\mathsf{E_{pt}}$ (e.g., using Algorithm 1)

4 Set $\mathsf{R := kP_t}$ on $\mathsf{E_t}$ (e.g., using Algorithm 1)

5 If $\mathsf{R \neq Q}$ mod t then **output** "failure".

6 Else **output** Q on $\mathsf{E_p}$

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] or [Kob91]. We also present a detailed analysis for this task in Appendix A.

4.1 Analysis of the Countermeasure.

We first show that Algorithm 8 computes the correct result if no error occurs. Algorithm 8 uses E_t to check the result of the scalar multiplication from Line 3. If $Q = kP$ on E_{pt}, then $Q = kP$ on E_t as well. This is evident from modular arithmetic. Note that any of the three exceptional cases in the addition formula ($P_1, P_2 = \mathcal{O}$ or $P_1 = -P_2$) can only occur if any of the intermediate results or the final result is equal to $\mathcal{O}$. However, $Q_i = \mathcal{O}$ on E_{pt} implies that $Q_i = \mathcal{O}$ on both E_p and E_t. As $k < \#E_p$, this may never happen. Hence, if no error occurred, Q is a valid point on both E_p and E_t. Given this fact, it is straightforward to see that $R = Q$ on E_t.

It remains to show that the proposed algorithm is secure against known Fault Attacks. 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 counteract fault attacks with random faults induced into any of the parameters used in Line 3 of Algorithm 8, we implement the countermeasure proposed by [BMM00] and [CJ03]. It requires checking whether the result of a scalar multiplication is a valid point on the original curve before returning a value. This check 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. Our analysis proves security against Sign Change Faults 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.

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 attacks, as only correlated attacks in the same run of Algorithm 8 would yield an advantage for an adversary. However, 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 $\mathsf{R} = \mathsf{Q}$ 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 $r_t \not| \kappa_i$, 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 $r_t \not| \kappa_i$, 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 (3), we have

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

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

$$\kappa_u = -2^{u+2} \sum_{j=u+1}^{n-1} k_j 2^{j-u-1} = -2^{i+2} \sum_{j=i+1}^{n-1} k_j 2^{j-i-1} - 2^{u+2} \sum_{j=u+1}^{i} k_j 2^{j-u-1}$$

$$= \kappa_i - 2^{u+2} \cdot \sigma_u, \quad \text{where } \sigma_u := \sum_{j=u+1}^{i} k_j 2^{j-u-1}, \tag{5}$$

and for $u > i$, we have

$$\kappa_u = -2^{u+2} \sum_{j=u+1}^{n-1} k_j 2^{j-u-1} = \kappa_i + 2^{i+2} \cdot \rho_u, \text{ where } \rho_u := \sum_{j=i+1}^{u} k_j 2^{j-i-1}. \tag{6}$$

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 (5) is not equal to 0 over the integers, it absolute value must be at least as large as r_t in order to be a multiple of r_t.

Exactly the same consideration holds for ρ_u. Both $|\sigma_u|$ and $|\rho_u|$ are smaller than $2^{|u-i|}$. Therefore, we must have $|u-i| > \lfloor \log(r_t) \rfloor$ in order to have a chance that the sums $|\sigma_u|$ and $|\rho_u|$ are larger than or equal to r_t. Otherwise, we cannot have $r_t \mid \kappa_i$ and $r_t \mid \kappa_u$.

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.

It has been noted in the literature that it is desirable to eliminate single points of failure ([YKLM03], [BOS03]). The explicit check proposed in Line 5 uses the zero flag as a single point of failure. However, one could easily modify the algorithm to use "infective computations" as defined in [YKLM03]. Lines 4, 5 and 6 would be replaced by

4 Set $\mathsf{R} := (1/\mathsf{k} \bmod \mathsf{r_t}) \cdot \mathsf{Q} - \mathsf{P}$ on $\mathsf{E_t}$ (using Algorithm 1)
5 Set $\mathsf{c} := \mathsf{R}[\mathsf{x}] + \mathsf{R}[\mathsf{y}]$
6 Set $\mathsf{S} := \mathsf{cQ}$ on $\mathsf{E_p}$ (using Algorithm 1)
7 **Output** S

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.

We first show that any NAF based variant of the repeated doubling algorithm is susceptible to Sign Change Faults. As noted before, these variants are often used in modern smartcards.

The most direct attack on the NAF is to induce a SCF in one of the signed digits of the recoded scalar k. I.e., change of -1 to $+1$ and vice versa, whereas a 0 is not changed at all. Although there exist compact representations of signed-digit representations, cf. [JT01a], the most popular encoding described in [JY00] admits a straightforward bit-flip attack. Note that an attacker being able to carry out the attacks described in [SA02] or [QS02] has no difficulties to induce the formerly sketched SCF on the recoded k.

Another conceivable 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], [AK96], [AK97], [KK99]). 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. This results in a transient Sign Change Fault induced into P as described in Section 3.2. 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.

Besides attacking the scalar multiplication algorithm on a high level, Sign Change Faults can also be realized by attacking the addition and multiplication procedures, which are used to compute point coordinates. One practical scenario for this attack relies on the internal functionality of many real-world embedded crypto co-processors supporting modular arithmetic, cf. [HP98] and [BOPV03]. Namely, in order to speed up the time-critical modular *multiplication* operation, they most often also rely on the non-adjacent form, cf. [Boo51], [Mac61], and [Kor93]. For practicability issues most often only the classical 2-NAF recoding is used. In this setting, also NAFs are used. However, this time, it is not the secret scalar k, which is transformed into NAF, but factors used during the computation of $P_1 + P_2$ for two elliptic curve point P_1 and P_2 (see below). Therefore, this attack also applies to variants of the repeated doubling algorithm, which are not based on the NAF of the secret scalar multiplier k.

The crucial point that we are exploiting here is that any efficient implementation of the NAF must provide special hardware to handle negative numbers, i.e., being able to compute the 2th complement of any register used as a multiplicand without time delay. Actually, considering the prototypical architecture of such an embedded crypto-co-processor, cf. [BOPV03] and [Sed87], this task is trivial to solve. Namely, simply invert every bit send to the long integer arithmetic unit (ALU) and additionally add $+1$. Note that providing this functionality is a must for the modulus register in order to implement the underlying modular multiplication algorithm efficiently, cf. [BOPV03], [Sed87], [WQ90], [Bar96] or [Mon85]. Given this functionality, it can be used for an attack.

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 2th 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 2th complement of any operand. Here, a fault attack can be mounted that results in an SCF.

For concreteness, let us consider the projective addition formulas, cf. [BSS99], [IEE98], for points $P_0 = (X_0 : Y_0 : Z_0)$, $P_1 = (X_1 : Y_1 : Z_1)$:

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

and for doubling the point $P_1 = (X_1 : Y_1 : Z_1)$:

$$\begin{array}{lll} M := 3X_1^2 + A Z_1^4, & Z_2 := 2Y_1 Z_1, & S := 4X_1 Y_1^2, \\ X_2 := M^2 - 2S, & T := 8Y_1^4, & Y_2 := M(S - X_2) - T. \end{array}$$

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, S_1, Z_2, or T. 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.

All attack settings introduced above can be further relaxed, taking into account that the device will check whether the final output is a valid point on the curve or not to defend against attacks described in [BMM00] and [CJ03]. In this case, attacks which change single bits with an acceptably high probability, e.g., byte faults, can be used, because unsuccessful attacks are winnowed by this final check.

6 Conclusions and Open Problems

The Sign Change Attack has been shown to be a practical attack 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.

[AK97] R. J. Anderson and M. G. Kuhn, *Low cost attacks on tamper resistant devices*, Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, LNCS, vol. 1361, Springer-Verlag, 1997, pp. 125–136.

[Bar96] P. Barrett, *Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor*, CRYPTO'86, LNCS, vol. 263, Springer-Verlag, 1996, pp. 311–323.

[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, Report 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.

[BOPV03] L. Batina, S. B. Ors, B. Preneel, and J. Vandewalle, *Hardware architectures for public key cryptography*, The VLSI Journal (2003), no. 34, 1–64.

[BOS03] J. Blömer, M. Otto, and J.-P. Seifert, *A new CRT-RSA algorithm secure against bellcore attacks*, CCS 2003, ACM SIGSAC, ACM Press, 2003, pp. 311–320.

[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, Report 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.

[Deu41] M. Deuring, *Die Typen der Multiplikatorenringe elliptischer Funktionenkörper*, Abh. Math. Sem. Hansischen Univ. **41** (1941), 197–272.

[GG99] J. von zur Gathen and J. Gerhard, *Modern computer algebra*, Cambridge University Press, 1999.

[GK86] S. Goldwasser and J. Kilian, *Almost all primes can be quickly certified*, Proc. 18th STOC (1986), 316–329.

[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.

[IT03] T. Izu and T. Takagi, *Exceptional procedure attack on elliptic curve cryptosystems*, PKC 2003, LNCS, vol. 2567, Springer-Verlag, 2003, pp. 224–239.

[JT01a] M. Joye and C. Tymen, *Compact encoding of non-adjacent forms with applications to elliptic curve cryptography*, CHES 2001, LNCS, vol. 2162, Springer-Verlag, 2001, pp. 377–390.

[JT01b] 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.

[KK99] O. Kömmerling and M. G. Kuhn, *Design principles for tamper-resistant smartcard processors*, Proceedings of the USENIX Workshop on Smartcard Technology — Smartcard '99, USENIX Association, 1999, pp. 9–20.

[Kob91] N. Koblitz, *Constructing elliptic curve cryptosystems in characteristic 2*, CRYPTO'90, LNCS, vol. 537, Springer-Verlag, 1991, pp. 156–167.

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

[Len87] H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Annals of Mathematics **126** (1987), 649–673.

[Mac61] O. L. MacSorley, *High-speed arithmetic in binary computers*, Proceedings of the IRE **49** (1961), no. 1, 67–91.
[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.
[OT04] K. Okeya and T. Takagi, *SCA-resistant and fast elliptic scalar multiplication based on wNAF*, IEICE Trans. Fundamentals **E87-A** (2004), no. 1, 75–84.
[QS02] J.-J. Quisquater and D. Samyde, *Eddy current for magnetic analysis with active sensor*, Proceedings of Esmart 2002, 2002.
[Rei60] G. W. Reitwiesner, *Binary arithmetic*, Advances in Computers (1960), no. 1, 231–308.
[SA02] S. Skorobogatov and R. Anderson, *Optical fault induction attacks*, CHES 2002, LNCS, vol. 2523, Springer-Verlag, 2002, pp. 2–12.
[Sch95] R. Schoof, *Counting points on elliptic curves over finite fields*, J. Théorie des Nombres de Bordeaux (1995), no. 7, 219–254.
[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.
[Sil00] J. H. Silverman, *The arithmetic of elliptic curves*, Graduate Texts in Mathematics, vol. 109, Springer-Verlag, 2000.
[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.
[YKLM03] S.-M. Yen, S. Kim, S. Lim, and S.-J. Moon, *RSA Speedup with Chinese Remainder Theorem Immune against Hardware Fault Cryptanalysis*, IEEE Transactions on Computers **52** (2003), no. 4, 461–472.

A On Choosing E_t with Prime Order

For the security of our countermeasure against Sign Change Attacks, we assume that both E_p and E_t have prime order. We assume that E_p and P are chosen first, which allows to use recommended curves (e.g., by [SEC00]). Afterwards, we choose E_t by choosing a triplet $(A_t, x_t, y_t) \in \mathbb{Z}_t^3$ uniformly at random until the curve E_t defined by that triplet has prime order. The order of such a curve E_t can be determined in polynomial time using Schoof's Algorithm [Sch95]. This curve E_t with $B_t := y_t^2 - x_t^3 - A_t x_t \bmod t$ and the point $P_t = (x_t : y_t : 1)$ is used to compute the combined curve E_{pt} and to check the result in Line 5 of Algorithm 8. The security analysis has shown 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. A minimal size for the order of P_t on E_t has been specified in Section 4.1.

Based on the following widely accepted conjecture, choosing $\#E_t$ as a prime is practical. Our conjecture is implied by the well-known Cramer's conjecture, stating that $\pi(x + \log^2(x)) - \pi(x) > 0$ for x sufficiently large [GK86]. Here, $\pi(x)$ is the prime counting function. Our conjecture has been introduced in a similar setting by Goldwasser and Kilian [GK86].

Conjecture 11 (Number of Primes in the Hasse Interval) *There exist constants $c_1, c_2 > 0$ such that*

$$\pi(t + 2\sqrt{t}) - \pi(t - 2\sqrt{t}) \geq \frac{c_2\sqrt{t}}{\log^{c_1}(t)}.$$

The following theorem states that we can find a small curve E_t with prime order efficiently if Conjecture 11 holds.

Theorem 12 (Choosing E_t with Prime Order) *Let $(A_t, x_t, y_t) \in \mathbb{Z}_t^3$ be chosen uniformly at random. (A_t, x_t, y_t) defines the curve E_t as $E_t : y_t^2 z_t \equiv x_t^2 + A_t x_t z_t^2 + B_t z_t^3 \bmod t$ where $B_t := y_t^2 - x_t^3 - A_t x_t \bmod t$. If Conjecture 11 is true, then there exists a constant $c > 0$ such that the probability that E_t defined by $(A_t, x_t, y_t) \in \mathbb{Z}_t^3$ has prime order is at least*

$$\frac{c \cdot c_2}{\log^{1+c_1}(t)},$$

where c_1, c_2 are as in Conjecture 11.

Proof. Given a triplet $(A_t, x_t, y_t) \in \mathbb{F}_t^3$, we want to determine the probability that the curve E_t defined by this triplet has prime order. As the parameter A_t is given and x_t and y_t define a point $P = (x_t : y_t : 1)$ on the curve, E_t is uniquely defined. As $z_t = 1$, the missing curve parameter B_t can easily be computed as $B_t = y_t^2 - x_t^3 - A_t x_t \bmod t$.

E_t is a valid elliptic curve only if the discriminant is non-zero, i.e., if $\gcd(4A_t^3 + 27B_t^2, t) = 1$. The chance that a triplet yields this case is negligible: There are $t^2 - t$ different curves over $\mathbb{F}_t$, each of which contains at least $t - 2\sqrt{t}$ many points. As the curve is uniquely defined if A_t and any of its points P_t are fixed, there are at least $(t^2 - t) \cdot (t - 2\sqrt{t})$ many valid triplets. As there are t^3 triplets in $\mathbb{F}_t^3$ altogether, we have

$$\begin{aligned}\Pr[(A_t, x_t, y_t) \text{ yields a valid curve } E_t] &\geq \frac{(t^2 - t) \cdot (t - 2\sqrt{t})}{t^3} \\ &= 1 - \left(\frac{2}{\sqrt{t}} + \frac{1}{t} - \frac{1}{t\sqrt{t}}\right) \geq 1 - \frac{3}{\sqrt{t}}.\end{aligned}$$

As this probability is negligibly far from 1, we will assume that all t^3 triplets define valid curves for simplicity. By Hasse's Theorem, we know that all possible group orders lie in the interval $[t+1-2\sqrt{t}, t+1+2\sqrt{t}]$, cf. [Sil00]. The theorem of Deuring (cf. [Len87] or [Deu41]) states that there exists a constant $c' > 0$ such that there are at least $c \cdot (t\sqrt{t})/\log(t)$ many elliptic curves for every given group order. This shows that the group orders are almost evenly distributed over all possible curves up to a factor of $1/\log(t)$. Among the $4\sqrt{t} + 1$ possible group orders, there are

$\Delta\pi := \pi(t+1+2\sqrt{t}) - \pi(t+1-2\sqrt{t})$ prime group orders, where $\pi(x)$ is the prime counting function. Following Conjecture 11, we have $c_1, c_2 > 0$ such that $\Delta\pi \geq \frac{c_2\sqrt{t}}{\log^{c_1}(t)}$. Each of these curves is uniquely defined by its A parameter and any of its points, i.e., there are at least $t - 2\sqrt{t}$ many triplets that define any of these curves. This holds because each of these curves has at least $t+1-2\sqrt{t}$ many points and $\mathcal{O}$ cannot be chosen in (A_t, x_t, y_t). Therefore, the probability that a random triplet $(A_t, x_t, y_t) \in \mathbb{F}_t^3$ yields an elliptic curve with prime order is

$$\Pr[\#E_t \text{ is prime}] \geq \frac{\Delta\pi \cdot c' \cdot t\sqrt{t} \cdot (t - 2\sqrt{t})}{\log(t) \cdot t^3} \geq \frac{c'}{\log(t)} \cdot \frac{\sqrt{t} - 2}{t} \cdot \frac{c_2\sqrt{t}}{\log^{c_1}(t)}$$
$$= \frac{c' \cdot c_2}{\log^{1+c_1}(t)} \cdot \left(1 - \frac{2}{\sqrt{t}}\right) = \frac{c \cdot c_2}{\log^{1+c_1}(t)},$$

where for $t > 16$ we can choose $c = c'/2$. □