Differential Fault Attacks on Elliptic Curve Cryptosystems (Extended Abstract)

Ingrid Biehl[1], Bernd Meyer[2], and Volker Müller[3]

[1] University of Technology, Computer Science Department, Alexanderstraße 10, 64283 Darmstadt, Germany, biehl@informatik.tu-darmstadt.de

[2] Siemens AG, Corporate Technology, 81730 München, Germany, bernd.meyer@mchp.siemens.de

[3] Universitas Kristen Duta Wacana, Jl. Dr. Wahidin 5–19, Yogyakarta 55224, Indonesia, vmueller@ukdw.ac.id

Abstract. In this paper we extend the ideas for differential fault attacks on the RSA cryptosystem (see [4]) to schemes using elliptic curves. We present three different types of attacks that can be used to derive information about the secret key if bit errors can be inserted into the elliptic curve computations in a tamper-proof device. The effectiveness of the attacks was proven in a software simulation of the described ideas.

Key words: Elliptic Curve Cryptosystem, Differential Fault Attack.

1 Introduction

Elliptic curves have gained especially much attention in public key cryptography in the last few years. Standards for elliptic curve cryptosystems (ECC) and signature schemes were developed [7]. The security of ECC is usually based on the (expected) difficulty of the discrete logarithm problem in the group of points on an elliptic curve. In many practical applications of ECC the secret key (the solution to a discrete logarithm problem) is stored inside a *tamper-proof* device, usually a smart card. It is considered to be impossible to extract the key from the card without destroying the information. For security reasons the decryption or signing process is usually also done inside the card.

Three years ago a new kind of attack on smart card implementations of cryptosystems became public, the so called *differential fault attack (DFA)*, which has been successful in attacking RSA [4], DES [3], and even helps reverse-engineering unknown cryptosystems. The basic idea of DFA is the enforcement of bit errors into the decryption or signing process which is done inside the smart card. Then information on the secret key can leak out of the card. In RSA implementations for example this information can be used to factor the RSA modulus (at least with some non-negligible probability), which is equivalent to computing the secret RSA key. So far there is no method known to extend the ideas of [4] to cryptosystems based on the discrete logarithm problem over elliptic curves. In this paper we investigate how DFA techniques can be used to compute the secret key of an ECC smart card implementation. Our attacks can be used for elliptic curves defined over arbitrary finite fields.

We consider the following scenario: a cryptographically strong elliptic curve is publicly known as part of the public key. The secret key $d \in \mathbb{Z}$ is stored inside a tamper-proof device, unreadable for outside users. On input of some point P on the chosen elliptic curve, the device computes and outputs the point $d \cdot P$. We assume that we have access to the tamper-proof device such that we can compute $d \cdot P$ for arbitrary input points P.

The main common idea behind the attacks in Sect. 4 is the following: by inserting (in the first mentioned attack) or by disturbing the representation of a point by means of a random register fault we enforce the device to apply its point addition resp. multiplication algorithm to a value which is not a point on the given but on some different curve. It is a crucial observation as we will show in Sect. 3 that the result of this computation is a point on the new probably cryptographically less strong curve which can be exploited to compute d. Thus these attacks work by misusing the tamper-proof device to execute its computation steps on group structures not originally intended by the designer of the cryptosystem. Similar ideas have been previously described in [10] where small order subgroups in $(\mathbb{Z}/p\mathbb{Z})^*$ are exploited to compute part of the secret key and in [5] for attacks against identification schemes. It is shown in [5] how identification schemes can be used to prove knowledge of logarithms and roots which do not even exist in the subgroup where the cryptosystem should make its computations.

Moreover, we present a DFA-like attack in Sect. 5 which is similar to attacks against RSA in [4]. There so called *register faults* are used to attack RSA smart card implementations. Register faults are transient faults that affect current data inside a register. All the circuitry is not influenced by these faults and works properly. For a more detailed discussion of that fault model, we refer to [4, Sect. 3]. We use the same fault model and assume that we can enforce random register faults in the decryption or signing process. Incorrect output values caused by random register faults are used to compute possible intermediate values of the computation and parts of the secret key. The intermediate values are not necessarily unique and one has to repeat the attack to get successively all bits of the secret key. The analysis of the probability of non-uniqueness and so of the costs of the computation of the secret key is the technically most complicated part of the analysis in the considered ECC case and cannot be based on the ideas presented in [4]. We sketch it in the appendix.

We know no widespread applications of smart cards for signature generation or decryption where complete points are the output of the used tamper-proof device. Therefore, we consider additionally as a more realistic scenario the situation that the tamper-proof device implements El-Gamal decryption. For El-Gamal decryption we can show that the attacks from Sect. 4.1 and 5 have expected polynomial running time. Furthermore, it is shown that the attack of Sect. 4.2

can be used against El-Gamal decryption and the elliptic curve digital signature scheme in expected subexponential running time.

The fault models of DFA attacks have been criticized for being purely theoretical. In [2] it is argued that a random one-bit error would be more likely to crash the processor of the tamper-proof device or yield an uninformative error than to produce a faulty ciphertext. Instead, *glitch attacks* which have already been used in the pay-TV hacking community, are presented in [2,1,8] as a more practical approach for differential fault analysis. The attacker applies a rapid transient in the clock or the power supply of the chip. Due to different delays in various signal paths, this affects only some signals and by varying the parameters of the attack, the CPU can be made to execute a number of wrong instructions. By carefully choosing the timing of the glitch, the attacker can possibly enforce register faults in the decryption or signing process and apply our attacks.

The paper is structured as follows: Section 2 gives an introduction to the well known theory of elliptic curves. Section 3 examines *pseudo-addition*, an operation which will play a crucial part in the DFA attacks. Sections 4 and 5 describe three different attacks on ECC systems and show how faults can be used to determine the secret key d. We close with comments on possible countermeasures.

2 Elliptic Curves

In this section we review several well known facts about elliptic curves. Let K be a finite field of arbitrary characteristic, and let $a_1, a_2, a_3, a_4, a_6 \in K$ be elements such that the discriminant of the polynomial given in (1) is not zero (the formula for the discriminant can be found in, e.g., [6]). Then the group of points $E(K)$ on the elliptic curve $E = (a_1, a_2, a_3, a_4, a_6)$ is given as

$$\Big\{(x, y) \in K^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\Big\} \cup \Big\{\mathcal{O}\Big\}, \tag{1}$$

where $\mathcal{O} := (\infty, \infty)$. Pairs of elements of K^2 which satisfy the polynomial equation (1) are denoted as *points on E*. In the following we use subscripts like P_E to show that P is a point on the elliptic curve E. We define the following operation:

– for all $P_E \in E(K)$, set $P_E + \mathcal{O}_E = \mathcal{O}_E + P_E := P_E$, (2)
– for $P_E = (x, y)_E$, set $-P_E := (x, -y - a_1x - a_3)_E$,
– for $x_1 = x_2$ and $y_2 = -y_1 - a_1\,x_1 - a_3$, set $(x_1, y_1) + (x_2, y_2) := \mathcal{O}_E$,
– in all other situations, set $(x_1, y_1)_E + (x_2, y_2)_E := (x_3, y_3)_E$, where

$$\begin{aligned} x_3 &= \lambda^2 + a_1\,\lambda - a_2 - x_1 - x_2 \\ y_3 &= -y_1 - (x_3 - x_1)\,\lambda - a_1\,x_3 - a_3 \end{aligned}$$

with

$$\lambda = \begin{cases} \dfrac{3\,x_1^2 + 2\,a_2x_1 + a_4 - a_1y_1}{2\,y_1 + a_1x_1 + a_3} & \text{if } x_1 = x_2 \text{ and } y_1 = y_2, \\[2ex] \dfrac{y_1 - y_2}{x_1 - x_2} & \text{otherwise.} \end{cases}$$

As shown in [6], this operation makes $E(K)$ to an abelian (additive) group with zero element $\mathcal{O}_E$. For any positive integer m we define $m \cdot P_E$ to be the result of adding P_E $m - 1$ times to itself. A crucial point that we will use in further sections is the fact that the curve coefficient a_6 is not used in any of the addition formulas given above, but follows implicitly from the fact that the point P_E is assumed to be on the curve E.

In almost all practical ECC systems the discrete logarithm (DL) problem in the group of points on an elliptic curve is used as a trapdoor one-way function. The DL problem is defined as follows: given an elliptic curve E and two points P_E, $d \cdot P_E$ on E, compute the minimal positive multiplier d. A *cryptographically strong elliptic curve* is an elliptic curve such that the discrete logarithm problem in the group of points is expected (up to current knowledge) to be difficult. ECC system implementations should always use cryptographically strong curves.

We will show in the following sections that random register faults can be used to compute information about a secret key d which is stored inside a tamper-proof device that computes $d \cdot P$ for some input point P. Thus our scenario becomes applicable if the device is used for the computation of the trapdoor one-way function $d \cdot P$ in a larger protocol. In practice however neither EC signature generation nor EC cryptosystems use tamper-proof devices which output complete points. Consider for example the following EC El-Gamal cryptosystem (without point compression):

Let E be a cryptographically strong elliptic curve. Given a point $P \in E$ assume that $Q = d \cdot P$ is the public key and $1 \leq d < \mathrm{ord}(P)$ the secret key of some user. For a point R let $x(R)$ denote the x-coordinate. The EC El-Gamal cryptosystem (without point compression) is given as follows:

Encryption
Input: message m, public key
choose $1 < k < \mathrm{ord}(P)$ randomly return $(k \cdot P,\ x(k \cdot Q) \oplus m)$

Decryption
Input: (H, m'), secret key d
compute $d \cdot H$ return $m' \oplus x(d \cdot H)$

If we combine the input and the output of the decryption process, then we can consider El-Gamal decryption as a black box that computes on input of some point H the x-coordinate of $d \cdot H$. Using the curve equation corresponding to the input point H we can determine the points $d \cdot H$ and $-(d \cdot H)$. But we have to stress that one cannot distinguish which one of this pair of points is $d \cdot H$.

3 Pseudo-addition and Pseudo-multiplication

Let E be a fixed cryptographically strong elliptic curve defined over a finite field K. We start with the following question: what happens when we use the operation defined in (2) for arbitrary pairs in K^2 instead for points on E? In this section we will answer this question and deduce some properties of this new operation.

Let $a_1, a_2, a_3, a_4 \in K$ be the coefficients of E with the exception of a_6. It should be noted that a_6 does not occur in the addition formulas (2) and is therefore not needed. Then it is easy to see that the operation (2) is also well-defined for arbitrary elements in $\mathcal{P} := K^2 \cup \{(\infty, \infty)\}$ (assuming that division by zero has the result ∞). For two arbitrary pairs $P_i \in \mathcal{P}$, $i = 1, 2$, we denote this operation as *pseudo-addition* and write $P_1 \oplus P_2$. *Pseudo-subtraction* is defined as pseudo-addition with the negative point and denoted with $P_1 \ominus P_2 = P_1 \oplus (-P_2)$. Moreover, for any positive integer $n \in \mathbb{N}$ and any pair $P_1 \in \mathcal{P}$, we define a *pseudo-multiplication* $n \otimes P_1$ as the result of $(\cdots((P_1 \oplus P_1) \oplus P_1) \oplus \cdots) \oplus P_1$, where pseudo-addition $\oplus$ is used exactly $n - 1$ times.

We present a few facts on the operation $\oplus$. Testing a few random example pairs in $\mathcal{P}$, it becomes obvious that pseudo-addition $\oplus$ is in general no longer associative. We can however prove the following weaker results on pseudo-addition.

Theorem 1. *Let two elements $(x_i, y_i) \in \mathcal{P}$, $i = 1, 2$, be given. Pseudo-addition is*

1. *commutative, i.e.* $(x_1, y_1) \oplus (x_2, y_2) = (x_2, y_2) \oplus (x_1, y_1)$,
2. *"weakly associative": if* $x_1 \neq x_2$ *or* $(x_1, y_1) = \pm(x_2, y_2)$

$$\Big((x_1, y_1) \oplus (x_2, y_2)\Big) \ominus (x_2, y_2) = (x_1, y_1).$$

Proof. The first assertion of the theorem follows directly from the symmetry of the formulas given in (2), testing all cases for the second assertion is a minor exercise for a computer algebra system. □

The discrete logarithm problem for elliptic curves is defined after multiplication of a point with a scalar. The following theorem describes a property of pseudo-multiplication.

Theorem 2. *Let the number of elements in the field K be q. For at least $q^2 + 1 - 4q$ elements $P \in \mathcal{P}$ and all positive integers n, m, pseudo-multiplication satisfies*

1. $n \otimes (m \otimes P) = (n \cdot m) \otimes P$,
2. $(n \otimes P) \oplus (m \otimes P) = (n + m) \otimes P$.

Proof. Note first that the assertions are trivial for the pair $\mathcal{O}$. Let therefore $P = (x, y) \in \mathcal{P}$. Define $a_6' = y^2 + a_1 xy + a_3 y - x^3 - a_2 x^2 - a_4 x$. If $(a_1, a_2, a_3, a_4, a_6')$ defines an elliptic curve, then obviously P is a point on this curve, and the result of the theorem follows directly from the associativity of point addition. The number of exceptional pairs (x, y) that do not lead to elliptic curves can easily be bounded by $4q$ since for given coefficients a_1, a_2, a_3, a_4 there are only two possibilities for a_6 such that the discriminant becomes zero. □

Finally, we examine how a fast multiplication algorithm behaves when used with pseudo-addition instead of ordinary point addition. A direct consequence of Theorem 2 is the following theorem.

Theorem 3. *Given a pair $P = (x, y) \in \mathcal{P}$ and a positive integer m. Assume that the tuple $(a_1, a_2, a_3, a_4, y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x)$ defines an elliptic curve E' over K. Then any fast multiplication type algorithm with input $(m, P, a_1, a_2, a_3, a_4)$ computes the result $m \otimes P$ accordingly to the addition defined in Sect. 2. Moreover, we have the equality $m \otimes P = m \cdot P_{E'}$, where $P_{E'} = P$ and $m \cdot P_{E'}$ are points on E' and the latter is computed with "ordinary" point additions.*

Remark 1. The crucial idea of pseudo addition is the fact that one of the curve coefficients is not used in the addition formulas. However a different point representation, so called *projective coordinates*, is also often used in practice. The addition formulas for such representations (see, e.g., [7, A.10.4]) have the same property. Therefore, the ideas presented in this paper can be adapted to other point representations typically used in practical applications.

4 Faults at the Beginning of the Multiplication

We start with the description of elliptic curve fault attacks. The first type of attacks however does not need the generation of any fault; it is an attack on "bad" implementations of ECC systems.

4.1 No Correctness Check for Input Points

The first attack is applicable when the device neither explicitly checks whether an input point P nor the result of the computation really is a point on the cryptographically strong elliptic curve E which is a parameter of the system. The attack is simple and should not be applicable to a well designed system, but nevertheless such a "bug" might happen in practice.

Let $E = (a_1, a_2, a_3, a_4, a_6)$ be a given cryptographically strong elliptic curve, which is part of the setup of the ECC system. In this situation we input a pair $P \in \mathcal{P}$ into the tamper-proof device which is not a point on E, but a point on some other elliptic curve E'. We choose the input pair $P = (x, y)$ carefully, such that with $a_6' = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x$ the tuple $(a_1, a_2, a_3, a_4, a_6')$ defines an elliptic curve E' whose order has a small divisor r and such that $\operatorname{ord}(P) = r$. With Theorem 3 we know that the output of the tamper-proof device with input P is then $d \cdot P$ on E'. Therefore, we end up with a discrete logarithm problem in the subgroup of order r generated by $P \in E'$, namely given points P, $d \cdot P$ on E', find $d \mod \operatorname{ord}(P)$. We can repeat this procedure with a different choice of P and use the Chinese Remainder Theorem to compute the correct value of d.

This algorithm is quite efficient if we do not choose P, but the curve E' first and compute P. The construction of such an elliptic curve E' can be done in essentially the same way as in the elliptic curve construction method described in [7]. First we try to find an integer m in the Hasse interval such that $(q + 1 - m)^2 - 4q$ has a large square factor and m a small factor. Then we can determine the j-invariant of an elliptic curve defined over K which has group order m. Finally, we have to check whether there exists an elliptic curve with coefficients $a_1, \dots, a_4, a'_6$ that has the given j-invariant. The latter test can be solved by factoring a polynomial of degree 2 and yields a'_6. We check for a few random values of x whether $y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a'_6 = 0$ is solvable for y. The pair $P_{E'} = (x, y)$ is chosen as input. Since m has a small divisor, given $d \cdot P_{E'}$ we can then determine the secret key modulo this small divisor (at least when this small divisor divides the order of $P_{E'}$ on E').

If we apply this attack to the device computing the El-Gamal decryption as described in Sect. 2 we cannot determine the y-coordinate of the resulting point uniquely. Given its x-coordinate w we can compute values z, z' such that $(w, z), (w, z') \in E'$ and $(w, z_1) = -(w, z_2)$, but we cannot decide which of these points is $d \cdot P$ on E'. By computation of the discrete logarithms of (w, z) and (w, z') we therefore get values c, c' with $c \equiv -c' \mod \operatorname{ord}(P)$ and either $d \equiv c \mod \operatorname{ord}(P)$ or $d \equiv c' \mod \operatorname{ord}(P)$. Thus we get $d^2 \equiv c^2 \mod \operatorname{ord}(P)$. To compute d we have to choose sufficiently many points P_i with small order such that $\operatorname{lcm}(\operatorname{ord}(P_1), \dots, \operatorname{ord}(P_s)) \geq d^2$. Then we get equations $d^2 \equiv c_i^2 \mod \operatorname{ord}(P_i)$ for $1 \leq i \leq s$ and can compute the value d^2 as an integer using the Chinese Remainder Theorem. The integer square root is the secret key d.

4.2 Placing Register Faults Properly

In the second attack we assume that we can enforce register faults inside the "tamper-proof" device at some precise moment at the beginning of the multiplication process. If the "tamper-proof" device checks whether the given input point is a point in the group of points of the cryptographically strong elliptic curve E, the attack of Sect. 4.1 is no more applicable. Assume however that we can produce one register fault inside the tamper-proof device right after this test is finished. Then the device computes internally with a pair P' which differs in exactly one bit from the input point P. Therefore, the device computes and – if it does not check whether the output is a point on E – outputs $d \otimes P'$. With Theorem 3 we deduce that $d \otimes P'$ lies on the same elliptic curve E' as P'. We determine a'_6 such that the output pair $d \otimes P'$ satisfies the curve equation with coefficients $(a_1, a_2, a_3, a_4, a'_6)$. If these coefficients define an elliptic curve E', we have reduced the original DL problem on E to a DL problem on E': check for all possible candidates P' (P' is unknown outside the device, but remember that P' differs in only one bit from the known point P) whether this candidate is a point on E' and – if so – try to solve the DL problem on E'. First, we compute $\operatorname{ord}(E')$ the number of points on E' using algorithms for point counting. If $\operatorname{ord}(E')$ has a small divisor r, we solve the DL problem for the points $(\operatorname{ord}(E')/r) \cdot P'_{E'}$ and $d \cdot ((\operatorname{ord}(E')/r) \cdot P'_{E'})$. This gives an equation $d \equiv c \mod r$ for some value c. Repeating this step with different divisors r we can compute d with the Chinese Remainder Theorem.

As described in Sect. 2, we can consider El-Gamal decryption as a black box that on input of some point P computes $x(d \cdot P)$ where d is the secret key stored inside the tamper-proof device. Note however that we cannot apply directly the attack from this section since we do not know the y-coordinate of the output point. Without the y-coordinate we cannot determine the curve E' to which the output P' belongs. In general there are many possible curves. It is however possible to solve the DL problem with non-negligible probability if there exists a curve E' corresponding to a base point P' resulting from a one-bit error such that the order of E' is smooth. Then we use the algorithm of Pohlig-Hellman (see [12]) to compute d.

Similar to the analysis of Lenstra's *Elliptic Curve Factoring Method* [9], it follows that we have to consider subexponentially many random elliptic curves until one of them has (subexponentially) smooth order. Thus the expected number of trials of the attack with random points $P \in E$ until we find such a smooth curve and can determine the secret multiplier d is subexponential again.

A similar situation occurs in the elliptic curve DSA signature scheme. In EC DSA, we have two primes p, q which are about the same size, an elliptic curve E over $\mathbb{F}_q$, and a point P on E of order p. The public key is (p, q, E, P, Q) where $Q = d \cdot P$ for some secret value d. To sign a message m with hash value h, the signer randomly chooses an integer $1 < k < p - 1$, computes $k \cdot P = (x_1, y_1)$, $r \equiv x_1 \mod p$, and $s \equiv k^{-1}(h + dr) \mod p$. The signature is (r, s).

Please note that we cannot input a point here but a publicly known point P is used as base point for the computation. We again disturb the computation of $k \cdot P$ by a register fault right at the beginning, i.e. P is replaced by some P'. The tamper-proof device then computes the signature $r' \equiv x(k \cdot P') \mod p$, and $s' \equiv k^{-1}(h + dr') \mod p$. Knowing this signature, we can use the following algorithm for all possible candidates $\tilde{P}$ for P':

- compute the curve $\tilde{E}$ corresponding to $\tilde{P}$ – if it exists –,
- derive from r' a small set of possible values for the x-coordinate of $x(k \cdot P')$ (since p, q are of about the same size),
- compute two candidates for the corresponding y-coordinate by means of the equation for $\tilde{E}$.

In case $\tilde{P}$ was correctly chosen and $\tilde{E}$ is a weak curve with respect to the discrete logarithm problem and $\mathrm{ord}(\tilde{P}) > p - 1$, one can first find k, and then the secret key d as $d \equiv r'^{-1}(s'k - h) \mod p$.

If we disturb the base point P in such a way that P' and P differ only in one bit, we have only $2 \log(q)$ possible choices for the curve E' and it is very unlikely that we get a curve with subexponentially smooth order and that the attack succeeds. But if we manage to change $o(\log(q))$ many bits at once such that we get subexponentially many different choices for E' then there is with high probability at least one curve with smooth order among them and we can compute the one-time key k and so the secret key d, i.e. the signature scheme is completely broken. The expected number of trials to get such a curve E' is subexponential again.

5 Faults at Random Moments of the Multiplication

In this section we sketch an attack that works even if we cannot influence the exact position in the computation process, at which the enforced random register fault happens.

In [4], the authors show how to attack RSA smart card implementations by enforcing register faults at random time in the decryption or signing process. The most important operation in RSA is fast exponentiation. For elliptic curves, the situation is similar and we can use some of the ideas of [4].

In the following we assume that the used elliptic curve is cryptographically strong, especially we assume that $E(\mathbb{F}_q)$ contains a subgroup of prime order p with $p > q/\log(q)$. The operation $Q = d \cdot P$ is usually done with either a "right-to-left" or a "left-to-right" multiplication algorithm. Since the ideas for the attacks in both cases are very similar we restrict ourselves here to the "right-to-left" multiplication algorithm and show: if one can enforce a fault randomly in a register at a random state of the computation than one can recover the secret key in expected polynomial time.

We start with a result for a fault model where we can introduce register faults during the computation of an a-priory chosen *specific block* of multiplier bits, e.g. we assume that we can repeatedly input some point P_E on E into the tamper-proof device and enforce a register fault during m successive iterations of the fast multiplication algorithm. Then we will show that we can relax this condition, i.e. even if one cannot influence at which block the register fault happens one can deduce the secret key after an expected number of polynomially many enforced random register faults. We will present a rather informal description of the attack which abstracts from some less important details.

The right-to-left multiplication algorithm works as follows (we denote by $(d_{n-1}\, d_{n-2}\, \ldots\, d_0)_2$ the binary representation of a positive integer d, where d_0 is the least significant bit):

```
H = P; Q = O;
for i = 0 , ... , n-1 do
   if (d_i == 1) then Q = Q + H;
   H = 2 * H;
output Q;
```

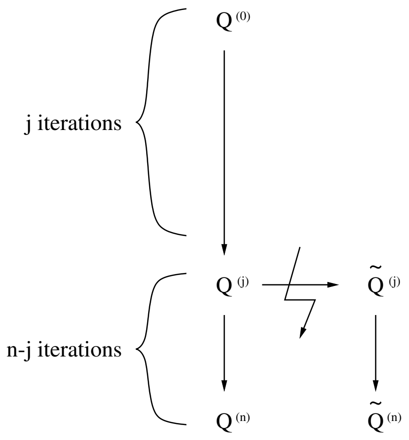
To simplify the notation assume that we know the binary length n of the unknown multiplier d (note that an attacker can "guess" the length of d). Denote by $Q^{(i)}$, $H^{(i)}$ the value stored in the variable Q, H in the algorithm description before iteration i.

The basic attack operation works as follows: we use the tamper-proof device with some input point P_E to get the correct result $Q^{(n)} = d \cdot P_E$ and moreover we restart it with input P_E but enforce a random register fault to get a faulty result $\tilde{Q}^{(n)}$. Assume that we enforce the register fault in iteration $n-m \leq j < n$, and that this fault flips one bit in a register holding the variable Q (the case that a bit in H is flipped can be handled similarly). Then $\tilde{Q}^{(j)}$ is a *disturbed Q-value*, i.e. a pair in $\mathcal{P}^2$ that differs in exactly one bit from $Q^{(j)}$.

Next we try to find the index of the first iteration j' with $j' > j$ and $d_{j'} = 1$ given $Q^{(n)}$ and $\tilde{Q}^{(n)}$. For simplicity reasons we assume that there is at least one non-zero bit among the m most significant bits of d, i.e. j' exists (we omit the technically more difficult case of m zero bits here for reasons of readability). We can find a candidate for the disturbed Q-value $\tilde{Q}^{(j')}$ with the following method: successively, we check each i with $n - m \le i < n$ as candidate for j', each $x \in \{0,1\}^{n-i}$ with least significant bit 1 as candidate for the i most significant bits of d, and each $Q_x^{(i)} = Q^{(n)} - x \cdot 2^i \cdot P_E$ as candidate for $Q^{(j)}$. For each choice of x and i we consider all disturbed Q-values $\tilde{Q}_x^{(i)}$ which we can derive from $Q_x^{(i)}$ by flipping one bit. Then we check whether this may be the disturbed value which appeared in the device, i.e. we simulate the computation of the device, compute the corresponding result value and check whether it is identical with the found value $\tilde{Q}^{(n)}$. More precisely: we use pseudo-additions with points $x_\ell 2^{i+\ell} \cdot P_E$ for $\ell = 0, \ldots, n-i-1$ where $x = (x_{n-i-1} \ldots x_0)_2$ with $x_0 = 1$ is the binary representation of x to get for candidates $i, x, Q_x^{(i)}$, and $\tilde{Q}_x^{(i)}$ the corresponding faulty result

$$\tilde{Q}_x^{(n)} = (\cdots((\tilde{Q}_x^{(i)} \oplus x_0 2^i \cdot P_E) \oplus x_1 2^{i+1} \cdot P_E) \oplus \cdots) \oplus x_{n-i-1} 2^{n-1} \cdot P_E .$$

If $\tilde{Q}_x^{(n)}$ is equal to the faulty result $\tilde{Q}^{(n)}$ output by the device, then we have found i as a candidate for j', $\tilde{Q}_x^{(i)}$ as a candidate for $\tilde{Q}^{(j')}$, and the binary representation of x as a candidate for the upper $n - j'$ bits of d.



By trying faults on Q and on H and all m possibilities for i and corresponding integers x we can make sure that this procedure outputs at least one candidate for $\tilde{Q}^{(j')}$ (or for $\tilde{H}^{(j)}$, in the case the fault occurs in H). In case there is only one candidate suitable for P_E, $Q^{(n)}$, m, and for $\tilde{Q}^{(n)}$ we have computed the $n - j'$ upper bits of the secret key d. One can show that the probability is small that more than one candidate survives (more details can be found in the appendix).

To reveal step by step all bits of d we start to compute the most significant bits as explained above and work downwards to the least significant bits by iterating the same procedure with new random register faults in blocks of at most m iterations. In each step we use the information that we already know about d to restrict the range of test integers x which have to be considered.

Theorem 4. *Let $m = o(\log\log\log q)$ and let n be the binary length of the secret multiplier. Assume that we can generate a register fault in an a-priory chosen block of m iterations of the multiplication algorithm. Using an expected number of $O(n)$ register faults we can determine the secret key d in expected $O(nm2^m(\log q)^3)$ bit operations.*

Finally, we consider the more general situation in which we cannot induce register faults in small blocks, but only at random moments during the multiplication. As in [4] one can show that for a large enough number ℓ of disturbed computations we get a reasonable probability that errors happen in each block of m iterations.

Theorem 5. *Let E be an elliptic curve defined over a finite field with q elements, let $m = o(\log\log\log q)$, and let n be the binary length of the secret multiplier. Given $\ell = O((n/m)\log(n))$ faults, the secret key can be extracted from a device implementing the "right-to-left" multiplication algorithm in expected $O(n\,2^m\,(\log q)^3\,\log(n))$ bit operations.*

Thus this theorem can be summarized as follows: if we consider the size of the used finite field as a constant then we need $O(n\log(n))$ accesses to the tamper-proof device to compute in $O(n\log(n))$ bit operations the secret key of bit length n. Please notice that the block size m we used as parameter of our algorithm reflects the tradeoff between the number of necessary register faults and the running time to analyse the output values influenced by these faults. It depends on the attackers situation whether more accesses to the tamper-proof device or more time for the analysis can be spent.

Remark 2. We have implemented a software simulation of the algorithm given above and attacked several hundred randomly chosen elliptic curves. Obviously, one can find easily non-unique solutions for the indices j and the parts of x of the corresponding discrete logarithms if the order of the base point P_E is small in comparison with 2^m where m is the length of the block containing the error. Also, if the size of the field is very small (< 1000) the algorithm often finds contradicting solutions. Both cases are not relevant for a cryptographically strong elliptic curve. In all tested examples with size of the field bigger than 2^{64}, randomly chosen curve, and random point on the curve we determined the complete secret multiplier d without problems.

If we apply this attack to the device computing the El-Gamal decryption as described in Sect. 2 we cannot determine the y-coordinate of the resulting point uniquely. Since we know the equation of the curve we can compute points Q and $-Q$ such that the correct result $Q^{(n)}$ of the device is one of these points. We start the described attack on both points Q and $-Q$ and compare only the x-coordinate of the disturbed results of the attack with the x-coordinate of the faulty result $x(\tilde{Q}^{(n)})$ of the device. Using this procedure we find at least one candidate for some point $\tilde{Q}^{(j)}$ (or for some point $\tilde{H}^{(j)}$, in the case the fault occurs in H) and can determine the upper bits of the secret multiplier d if the candidate is unique.

6 Countermeasures

It became obvious in the preceding sections that DFA techniques for elliptic curves depend mainly on the ability to disturb a point on E to "leave" the group of points and become an ordinary pair in $\mathcal{P}$. Countermeasures against all attacks presented in this paper are therefore obvious. Although it is part of the protocols of most cryptosystems based on elliptic curves to check whether input points indeed belong to a given cryptographically strong elliptic curve it follows from the described attacks that it is even more important for the tamper-proof device to check the output point or any point which serves as basis for the computation of some output values. If any of these points, input points or computed points, do not satisfy this condition, no output is allowed to leave the device. This countermeasure for ECC is similar to the countermeasures proposed against DFA for RSA where the consistency of the output also has to be checked by the device.

Acknowledgements

We would like to thank the unknown referees for several suggestions which improved the quality and readability of the paper. Moreover, we would like to thank Susanne Wetzel and Erwin Heß for discussions. Our thank belongs especially to Arjen K. Lenstra who gave us a lot of support to improve the paper and pointed out to us the subexponential time attacks against El-Gamal decryption and EC DSA in Sect. 4.2.

References

1. R. J. Anderson and M. G. Kuhn: *Tamper Resistance – a Cautionary Note*, Proceedings of Second USENIX Workshop on Electronic Commerce 1996, pp. 1–11.
2. R. J. Anderson and M. G. Kuhn: *Low Cost Attacks on Tamper Resistant Devices*, Lecture Notes in Computer Science 1361, Proceedings of International Workshop on Security Protocols 1997, Springer, pp. 125–136.
3. E. Biham and A. Shamir: *Differential Fault Analysis of Secret Key Cryptosystems*, Lecture Notes of Computer Science 1294, Proceedings of CRYPTO'97, Springer, pp. 513–525.
4. D. Boneh, R. A. DeMillo, and R. J. Lipton: *On the Importance of Checking Cryptographic Protocols for Faults*, Lecture Notes of Computer Science 1233, Proceedings of EUROCRYPT'97, Springer, pp. 37–51.
5. M. Burmester: *A Remark on the Efficiency of Identification Schemes*, Lecture Notes of Computer Science 473, Proceedings of EUROCRYPT'90, Springer, pp. 493–495.
6. I. Connell: *Elliptic Curve Handbook*, Preprint, 1996.

7. IEEE P1363 Draft Version 12: *Standard Specifications for Public Key Cryptography*, available on the Homepage of the IEEE.
8. O. Kömmerling and M. G. Kuhn: *Design Principles for Tamper-Resistant Smartcard Processors*, Proceedings of USENIX Workshop on Smartcard Technology 1999, pp. 9–20.
9. H. W. Lenstra: *Factoring Integers with Elliptic Curves*, Annals of Mathematics, **126** (1987), pp. 649–673.
10. C. H. Lim and P. J. Lee: *A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup*, Lecture Notes of Computer Science 1294, Proceedings of CRYPTO'97, Springer, pp. 249–263.
11. A. Menezes: *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.
12. S. Pohlig and M. Hellman: *An Improved Algorithm for Computing Logarithms over* GF(p) *and its Cryptographic Significance*, IEEE Transactions on Information Theory, vol. 24 (1978), pp. 106–110.
13. J. H. Silverman: *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics 106, Springer 1986.

Appendix: Success Probability of the Attack in Sect. 5

We denote by $Q^{(i)}$ resp. $H^{(i)}$ the value stored in the variable Q resp. H before iteration i of the right-to-left multiplication algorithm described in Sect. 5. We know also the correct result $Q^{(n)} = d \cdot P_E$ and a faulty result $\tilde{Q}^{(n)}$ for a given base point P_E on E.

We define a *disturbed Q-value* with respect to P_E, $Q^{(n)}$, m to be a pair in $\mathcal{P}^2$ that differs in exactly one bit from some $Q^{(i)}$ for $n - m \leq i \leq n$. Assume that we enforce a register fault in iteration $n - m \leq j < n$, and that this fault flips one bit in a register holding the variable Q. Denote by $\tilde{Q}^{(j)}$ the resulting disturbed Q-value. According to the right-to-left multiplication algorithm we try all possible indices $n - m \leq i < n$ and all integers x with exactly $n - i$ bits (least significant bit 1) to compute candidates $\tilde{Q}_x^{(i)}$ for disturbed Q-values that lead to the faulty result $\tilde{Q}^{(n)}$. The second place where a register fault can happen is the register holding the variable H in the algorithm. The procedure for this case is quite similar. Again, we try all possible indices $n - m \leq i < n$ and all integers x of exactly $n - i$ bits (least significant bit 1). If the fault is now introduced in the variable H (i.e. into one of the points $H^{(i)} = 2^i \cdot P_E$), this results in some disturbed H-value $\tilde{H}^{(i)}$ and is then propagated by the loop of the algorithm. By trying both Q- and H-case and all m possibilities for i and corresponding integers x we can make sure that this procedure outputs at least one candidate for $\tilde{Q}^{(j)}$ or for $\tilde{H}^{(j)}$. In case there is only one candidate suitable for P_E, $Q^{(n)}$, m and for $\tilde{Q}^{(n)}$ we call this candidate a *uniquely determined disturbed value* with respect to P_E, $Q^{(n)}$, m. Otherwise, a candidate is called *non-uniquely determined disturbed value.*

In Lemma 2 we will prove that for $m = o(\log \log \log q)$, all d and almost all points P_E there are at most three different non-uniquely disturbed values. Thus the expected number of necessary repetitions of attacks (i.e. choosing a point

P_E and causing a random register fault in the last m iterations), until one finds a uniquely determined disturbed value, is constant. Next we give an estimate for the probability that an attack allows us to find a uniquely determined disturbed value. For background on elliptic curve theory, we recommend [6] or [13].

Lemma 1. *Let $m = o(\log\log\log q)$ and assume that we can generate register faults in the last m iterations of the algorithm. The number of points P_E for which there exist more than three different non-uniquely determined disturbed values with respect to P_E, $Q^{(n)}$, m is bounded by $O((\log\log q)(\log q)^5)$.*

Proof. We want to bound the number of points P_E for which there exists at least four different non-uniquely determined disturbed values with respect to P_E, $Q^{(n)}$, m. Thus there are at least two pairs of disturbed values where each pair leads under the secret key d with $Q^{(n)} = d \cdot P_E$ to the same faulty multiplication result. Since these disturbed values are either H- or Q-values the following cases must be considered: each such pair either consists of two disturbed Q-values, or two disturbed H-values or is a pair consisting of one disturbed Q- and one disturbed H-value. We show for all nine cases that the number of points P_E for which there exists four different non-uniquely disturbed values can be bounded by $O((\log\log q)(\log q)^5)$.

We consider the first case that all four non-uniquely disturbed values are Q-values. Then there exist integers x_i of binary length at most m, points $P_i \in E(\mathbb{F}_q)$, and bit locations r_i for $1 \le i \le 4$ such that

1. $P_1 + x_1 \cdot R = P_2 + x_2 \cdot R = Q^{(n)}$,
2. $P_{1,(r_1)} \otimes x_1 R = P_{2,(r_2)} \otimes x_2 R$,
3. $P_3 + x_3 \cdot R = P_4 + x_4 \cdot R = Q^{(n)}$,
4. $P_{3,(r_3)} \otimes x_3 R = P_{4,(r_4)} \otimes x_4 R$,

where n is the binary length of the secret multiplier, $R = 2^{n-m} \cdot P_E$, $P_{i,(j)}$ denotes a pair which is obtained by switching bit j of point P_i (numbering the bits of x- and y-coordinate appropriately), and the notation $P \otimes w \cdot R$ serves as abbreviation for the computation $(\cdots((P \oplus w_0 \cdot R) \oplus w_1 \cdot 2 \cdot R) \oplus \cdots) \oplus w_{k-1} \cdot 2^{k-1} \cdot R$ for an integer $w = (w_{k-1} \ldots w_0)_2$. (The values $P_{i,(j)}$ are the non-uniquely determined disturbed values to the faulty results $P_{1,(r_1)} \otimes x_1 R$, and $P_{3,(r_3)} \otimes x_3 R$.)

We translate the four conditions above into polynomial equations using the concept of formal points. Assume that P_1 is given formally as (X_1, Y_1) and R as (X_2, Y_2). Using the theory of division polynomials (see [6]), it follows directly that the X_2-degree of the numerator, denominator, of points $x \cdot R$ for arbitrary m-bit integers x is $O(2^{2m})$. Combining the first and third equation (note that $Q^{(n)}$ occurs in both equations), we see with the addition formulas that the x-coordinates of all the points $P_i, i \ge 2$, can be written as rational functions of constant degree in X_1, Y_1, Y_2 and of degree $O(2^{cm})$ in X_2 for some small constant c (both numerator and denominator). The essentially same idea can be used to find an equation from the second and the fourth equation: we compute the left hand side as rational functions (using the representation of P_1, P_3 in X_1, Y_1, X_2, Y_2, respectively), introducing new variables for the faults r_1, r_3. Similarly, we transform the right hand side of the second and fourth equation into a rational function, introducing new variables for the faults r_2, r_4 and using the representation of P_2, P_4 as function in X_1, Y_1, X_2, Y_2. Then we can derive a polynomial of X_1, X_2-degree $O(2^{c'm})$ for some small constant c'. Using the fact that both P_1 and R are points on E, we can remove the variables Y_1, Y_2 with the help of the curve equation, increasing the exponent in the degree formula by a constant. Finally, we determine the resultant in the variable X_2 of both these equations, thereby removing X_2 and getting an equation of X_1-degree $O(2^{2^{c''m}})$ for some constant c'' (the resultant can be determined by computing the determinant of the so called Sylvester matrix). By substituting all possible values for $r_i, 1 \leq i \leq 4$, (note that r_i are bit faults) and substituting all possible values for $x_i, 1 \leq i \leq 4$, (note that $0 \leq x_i \leq 2^m$), and observing that $m = o(\log\log\log q)$ and so $O(2^{2^{c''m}}) = O(\log q)$, we get $O(2^{4m}(\log q)^4) = O((\log\log q)(\log q)^4)$ equations of X_1-degree $O(\log q)$ each. Therefore, the total number of possibilities for X_1 and the number of possible points P_E is at most $O((\log\log q)(\log q)^5)$.

The number of points P_E for the other cases can be analyzed analogously. □

Lemma 2. *Let $m = o(\log\log\log q)$ and q be sufficiently large. The expected number of attacks, i.e. random choices of a point P_E of $E(\mathbb{F}_q)$ and random register faults in the last m iterations of the right-to-left multiplication algorithm, until one finds a uniquely determined disturbed value, is 2.*

Proof. Since $E(\mathbb{F}_q)$ is cryptographically strong, it contains a subgroup of prime order p with $p > \frac{q}{\log q}$. Using the Hasse theorem it follows from the previous lemma that we will get a point P_E with probability $1 - c(\log\log q)(\log q)^5/q$ (for some constant c) of order at least p and for which there exists at most three different non-uniquely determined disturbed values with respect to P_E, $Q^{(n)}$, m. Since at least all $H^{(i)}$ for $i = n - m, \ldots, n$ are different and consist of $2\log q$ bits, there are $O(m \log q)$ bit positions in the computation process which could be disturbed. Since there are less than four non-uniquely determined disturbed values for P_E the probability to disturb the computation in a way which will lead to one of these non-uniquely determined disturbed values is bounded by $3/(m \log q)$. It follows that with probability more than 1/2 each attack will lead to a uniquely determined disturbed value. □

Lemma 3. *Let $m = o(\log\log\log q)$. Assume that we can generate random register faults in the last m iterations of the algorithm of the attack. Then the expected number of applications of the algorithm with independent random register faults is $O(m)$ until we can compute the m most significant bits of the secret key d. Thus the expected number of bit operations is $O(m^2 2^m (\log q)^3)$.*

Proof. For a running time analysis we note that the number of fault positions is at most 4 $\log q$ in each iteration (there are at most 2 points that can be disturbed, the x- and y-coordinate of each point have at most $\log q$ bits). For each of the 2^{m+1} different integers x we have at most m pseudo-additions which can be done in $O(m(\log q)^2)$ bit operations each. In addition we have to compute for all indices $n - m \leq j < n$ the corresponding values $Q^{(j)}$ and $H^{(j)}$ which can be done in $O((\log q)^3)$ bit operations.

We learn all m most significant bits of d if the error changes the value $H^{(n-m)}$. The probability that a random error during the last m iterations disturbs a bit of $H^{(n-m)}$ is $\frac{1}{2m}$. Therefore, we can lower bound the probability of success if we have k independent randomly disturbed results by $1 - (1 - \frac{1}{2m})^k$. Since the register faults are induced at random places, we derive that we expect to need $k = O(m)$ many faulty applications before we have found all the upper m bits of d. Combining all the partial results, we get the expected $O(m^2 2^m (\log q)^3)$ bit operations. □

Lemma 2 is the basis for an algorithm to determine the complete multiplier d. The basic idea is the usage of Lemma 2 successively on blocks of size m. Note the fact that we can "compute backwards" once we know the upper m bits of d to generate a DL problem with a smaller multiplier. Computing backwards from the correct output of the device for a given base point to get a correct intermediate point is trivial. For the disturbed output of the device it follows from Theorem 1 that we can compute backwards the faulty result with high probability too to get a faulty intermediate result. Then we can apply Lemma 2 again on the pair of correct intermediate point and faulty intermediate result. Thus we get:

Theorem 6. *Let $m = o(\log \log \log q)$ and let n be the binary length of the secret multiplier. Assume that we can generate a register fault in a block of m iterations of the right-to-left multiplication algorithm. Using an expected number of $O(n)$ register faults we can determine the secret key d in expected $O(nm2^m(\log q)^3)$ bit operations.*