Bit-Flip Faults on Elliptic Curve Base Fields, Revisited

Taechan Kim[1,⋆] and Mehdi Tibouchi[2]

[1] Seoul National University
[2] NTT Secure Platform Laboratories

Abstract. As part of their investigation of fault attacks on elliptic curve cryptosystems, Ciet and Joye showed, back in 2003, that perturbing the value representing the cardinality of the base field in a physical implementation of ECC could result in a partial key recovery. They had to assume, however, that the perturbed computation would "succeed" in some sense, and that is rather unlikely to happen in practice.

In this paper, we extend their analysis and show that, in a somewhat stronger fault model, full key recovery is possible with a single fault. For example, our fault attack typically reduces 256-bit ECDLP to solving discrete logarithm problems in a few random elliptic curves over fields of less than 60 bits, which typically takes a matter of seconds. More generally, the asymptotic complexity of ECDLP becomes heuristically subexponential under our fault attack.

Our attack also extends to a very efficient full key recovery attack on ECDSA with two faulty signatures.

Keywords: Elliptic Curve Cryptography, Fault Analysis, ECDSA.

1 Introduction

Elliptic Curve Cryptography. Elliptic curves, whose use in public-key cryptography was first suggested by Koblitz and Miller in the mid-1980s [29,31], offer numerous advantages over more traditional settings like RSA and finite field discrete logarithms, particularly higher efficiency and a much smaller key size that scales gracefully with security requirements. This makes them especially well-suited for implementations of cryptography, in both hardware and software, on resource-constrained and embedded devices.

Despite some initial reluctance from practitioners, elliptic curve cryptography (ECC) has become widely accepted in the cryptographic community and has made major inroads in the industry starting from the early 2000s, with the standardization of multiple elliptic curve-based cryptographic primitives [24,3] (ECDSA [18] in particular has been widely adopted), agencies weighing in favor of their use [32], and new embedded applications such as secure e-passports mandating them [25].

⋆ This work was carried out during the first author's visit to NTT Secure Platform Laboratories.

I. Boureanu, P. Owesarski, and S. Vaudenay (Eds.): ACNS 2014, LNCS 8479, pp. 163–180, 2014.

ECC implementations are now a staple of the modern cryptographic landscape, and it is thus of prime importance to study their security not only from a purely algorithmic point of view, but also against physical attacks. A number of works in recent years have been devoted to attacking ECC implementations with, and protecting them against, both side-channel analysis, fault analysis, and even combined attacks with both passive and active phases. See [14,2,15] for surveys of recent results.

Fault Attacks on Elliptic Curves. In this paper, we are particularly interested in *active* physical attacks against the elliptic curve discrete logarithm problem (ECDLP). Much like the so-called Bellcore attack of Boneh, DeMillo and Lipton [9] allows an attacker to recover an RSA secret key by perturbing the computation of an RSA signature in a signing device, fault attacks on ECDLP seek to recover a secret exponent k by perturbing the computation of the scalar multiplication of an elliptic curve point P by k.

The most common type of fault attack on ECDLP consists of "weak curve attacks", the first of which was introduced by Biehl, Meyer and Müller in 2000 [8]. Their key observation was that at least one of the elliptic curve parameters does not intervene in the expression of the addition and doubling formulas on that elliptic curve; for example, in commonly used coordinate systems, the addition and doubling formulas on the short Weierstrass curve $E\colon y^2 = x^3 + ax + b$ do not involve the parameter b at all. As a result, if one perturbs the computation of a scalar multiplication kP by modifying the coordinates of P into those of a different point $\widetilde{P}$ not on the curve E, the computation carried out by the device is still an elliptic curve scalar multiplication $k\widetilde{P}$, but on a different curve:

$$\widetilde{E}\colon y^2 = x^3 + ax + \tilde{b} \quad \text{where} \quad \tilde{b} = y_{\widetilde{P}}^2 - x_{\widetilde{P}}^3 - ax_{\widetilde{P}}.$$

If the discrete logarithm problem happens to be easy on $\widetilde{E}$ (this happens for example when the group order of $\widetilde{E}$ is a smooth number, or when $\widetilde{P}$ is a point of small order on $\widetilde{E}$), information about k can be deduced from the result $k\widetilde{P}$ of the faulty scalar multiplication.

This type of attack was later called *invalid point attack* because it relies on obtaining, through fault injection, an input point to the scalar multiplication algorithm that lies outside of the elliptic curve. One notable variant is the twist attack of Fouque et al. [19], which assumes that the input point is in compressed form, and which perturbs it (with probability 1/2) into a point on the twisted curve of E, which is often weaker.

Ciet and Joye [11] extended the work of Biehl et al. in several directions, and showed in particular that faults injected not only on the base point but also on the curve parameters, or the representation of the base field, could cause the scalar multiplication to be performed on a weak curve, and hence allow the recovery of some information on the discrete logarithm.

We consider in particular their attack on the base field. In the prime field case, it can be roughly described as follows: suppose that the elliptic curve E is defined over the field $\mathbb{F}_p$, and that a fault is injected on p, yielding a faulty modulus p'. With high probability, p' is then a composite number $p' = \prod_{i=1}^{r} q_i^{e_i}$. Hence, the faulty computation of kP, *assuming that it proceeds without error*, should really be a scalar multiplication by k on an elliptic curve over the ring $\mathbb{Z}/q_1^{e_1}\mathbb{Z} \times \cdots \times \mathbb{Z}/q_r^{e_r}\mathbb{Z}$, which decomposes as a product $\widetilde{E}_1(\mathbb{Z}/q_1^{e_1}\mathbb{Z}) \times \cdots \widetilde{E}_r(\mathbb{Z}/q_r^{e_r}\mathbb{Z})$. Each of the curves $\widetilde{E}_i$ is defined over a much smaller base than the original curve E, and is therefore weaker, making the discrete logarithm problem potentially tractable.

If one tries to implement that attack, however, one finds that it fails almost all the time, mainly because some of the prime factors q_i of p' are likely to be very small; the reduced curve modulo those primes may either not be an elliptic curve at all (due to having zero discriminant: this happens if one of the q_i's is 2 or 3, for example), or have such a small order that the scalar multiplication will meet the point at infinity modulo that prime with high probability, causing a division by zero or an otherwise erroneous computation.

Our Contributions. Motivated in part by recent work on modulus fault attacks against RSA signatures [10], we revisit Ciet and Joye's attack on the base field representation in such a way as to make it actually work with as few as a single faulty scalar multiplication.

Most of our analysis is in a stronger fault model than random fault model considered by Ciet and Joye: namely, we assume that the adversary can *choose* the fault they inject on the modulus from a certain set. This can be achieved in various ways in practice, such as instruction skipping in software or triggering reset wires in hardware to zero out some chosen bitstring in the modulus, or by using the chosen bit-flip model considered in the original paper of Biehl et al.. Bit flips are often considered tricky to achieve against real devices, but Agoyan et al. [1] have demonstrated that they can be performed in a reproducible way using laser shots on SRAM memory cells.

We show that if one can flip a chosen bit of the modulus p (or otherwise inject a fault chosen from a similarly-sized set), we can ensure that the resulting faulty modulus p' has much smaller prime factors (less than 60 bits each when p is a 256-bit prime, say). This makes it possible to recover almost all the bits of k (all but a couple dozen) from a *single* faulty computation of the scalar multiplication kP in a division-free coordinate system (projective, Jacobian, etc.). The few remaining bits can be quickly found using Pollard's lambda algorithm if, in addition, the result of the non-faulty computation is known. The attack also extends to the more common case when the result is converted back to affine coordinates at the end, unless the device implements a normally useless test to avoid it.

We provide an extensive theoretical analysis of this attack showing, more generally, that a chosen bit-flip on an n-bit elliptic curve modulus allows the recovery of $n - O(\log^2 n)$ bits of the n-bit exponent k from a faulty computation of kP in heuristic (slightly) subexponential time $2^{O(n \log\log n / \log n)}$. And again, all

n bits can be recovered if the correct value of kP is also known, making ECDLP subexponential under this fault model. Note that our fault is chosen from a set of polynomial size: this is in contrast with Biehl et al.'s subexponential complexity result for their invalid point attack (later established more rigorously under reasonable heuristics by Wang and Zhan [35]) which requires superpolynomially many faults, and is therefore of little practical significance.

Furthermore, our attack extends to random faults as well, but several faults are necessary for the recovery of the whole discrete log to become practical: for 256-bit elliptic curves, three or four faults are typically enough.

In addition, we propose a very efficient full key recovery attack on ECDSA with only two faulty signatures based on our attack: each faulty signature reveals most of the bits of the corresponding nonce, so that the ECDSA secret key can easily be found using lattice reduction techniques with just two faults. Moreover, for deterministic implementations of ECDSA, having one correct and one faulty signature on the same message is also enough.

All variants of our attacks have been validated using simulations. In our implementation, the search for the optimal bit flip on a 256-bit modulus takes a few CPU hours (and is easily parallelized) on a standard PC, while the attacks themselves, both on ECDLP and ECDSA, take seconds to minutes to complete.

2 Background on ECDLP and ECDSA

Elliptic Curves. An elliptic curve over a finite prime field $\mathbb{F}_p$, $p > 3$, can be described as the set of points (x, y) on the affine plane curve

$$E\colon y^2 = x^3 + ax + b \tag{1}$$

for some coefficients $a, b \in \mathbb{F}_p$ with $4a^3 + 27b^2 \neq 0$, together with the point at infinity on the projective closure of the affine curve. This set $E(\mathbb{F}_p)$ is endowed with a natural abelian group law which can be defined by a chord-and-tangent process.

The formulas giving the affine coordinates of the sum of two curve points involve divisions, so efficient implementations of elliptic curve arithmetic rely on redundant coordinate systems for which division-free formulas exist. In this paper, we consider in particular Jacobian coordinates $(X : Y : Z)$ for the Weierstrass form (1), given by $(x, y) = (X/Z^2, Y/Z^3)$ as they seem to be the most commonly used in practice (in particular, they are the coordinate system described in the IEEE P1363 standard [23, A.10], and the one recommended over prime fields in [22]). However, our results generalize directly to any other coordinate system with division-free formulas for addition and doubling which are independent of at least one curve parameter (like the formulas for Jacobian coordinates below, which are independent of the parameter b of the Weierstrass equation). This also includes projective coordinates in Weierstrass form, as well as projective coordinates on Hessian and generalized Hessian curves [16], Huff and twisted Huff curves [26], etc., but not Edwards curves [7], for which addition formulas depend on all curve parameters.

Formulas for addition and doubling in Jacobian coordinates are provided in the full version of this paper [28] (as well as many other standard references); they are indeed division-free and do not depend on the parameter b of the curve equation. Such formulas are sufficient to compute scalar multiplications $P \mapsto kP$ in $E(\mathbb{F}_p)$. While this can be done in several ways, we will assume in this paper that the double-and-add algorithm (in either its left-to-right or its right-to-left variant) used for those computations. Nevertheless, we note that our results apply with little to no change to most other scalar multiplication algorithms, including the generalized Montgomery ladder [27], and even to higher radix algorithms (k-ary double-and-add, sliding window, etc.) as long as faults are injected before the corresponding precomputations so that precomputed points lie on the faulty curve as well.

ECDLP. Consider again an elliptic curve E over $\mathbb{F}_p$, and a point $P \in E(\mathbb{F}_p)$ of order N in the group (usually a generator of either $E(\mathbb{F}_p)$ itself or a subgroup of small index). The elliptic curve discrete logarithm problem (ECDLP) in (the subgroup generated by P in) E is the computational problem of finding $k \in \{0, \ldots, N-1\}$ given P and the scalar multiple kP. For almost all isomorphism classes of elliptic curves, this problem is considered hard, and the best known attack has a complexity of $O(\sqrt{N})$.

ECDSA. The elliptic curve digital signature algorithm (ECDSA) is a digital signature scheme based on elliptic curves and standardized as part of the Digital Signature Standard [18]. ECDSA system parameters consist of an elliptic curve E over a finite field (for example a prime field $\mathbb{F}_p$), a generator $P \in E(\mathbb{F}_p)$ of a large subgroup of the curve, of order N, and a hash function $H\colon \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z}$. The secret key is then a random integer $\mathtt{d} \in \{0, \ldots, N-1\}$, and the public verification key is the curve point $\mathtt{d}P$. A message m is then signed as follows: choose a uniformly random $k \in \{1, \ldots, N-1\}$, compute the scalar multiple $kP = (x, y) \in E(\mathbb{F}_p)$, and return the signature as the pair $(r, s) \in (\mathbb{Z}/N\mathbb{Z})^2$ given by $r = x \bmod N$ and $s = k^{-1} \cdot \big(H(m) + r \cdot \mathtt{d}\big) \bmod N$.

3 Our Attack

3.1 Attack Model

We consider a cryptographic device computing the scalar multiplication of a known point $P = (x_P, y_P)$ by an unknown scalar k on a public elliptic curve $E\colon y^2 = x^3 + ax + b$ over the finite prime field $\mathbb{F}_p$, and we try to recover k using fault analysis. To fix ideas, we assume that the scalar multiplication is carried out in Jacobian coordinates using the double-and-add algorithm (although, as mentioned earlier, the approach extends to most curve shapes and coordinate systems admitting division-free addition and doubling formulas, and to essentially any scalar multiplication algorithm with minor changes).

Our fault model is to consider that a permanent fault is injected on the representation of the base field $\mathbb{F}_p$ at the beginning of the computation, so that all arithmetic operations are carried out modulo a different integer p' instead, which is typically composite: $p' = \prod_{i=1}^{r} q_i^{e_i}$. This type of fault is typically achieved using laser beams on RAM cells after the value of p is loaded into memory. Note that the model is only realistic if the device uses a generic implementation of base field operations. Devices using curves with random base fields, such as Brainpool [30] and the French government ANSSI curve [4], do satisfy that property, but in some cases implementations of NIST curves [18] (which have special base fields) do not, and those based on Curve25519 [5] and other "SafeCurves" [6] normally do not either.

As noted in the introduction, the computation of the scalar multiplication by k using the faulty modulus p' essentially amounts to a scalar multiplication by k in the product group $\widetilde{E}_1(\mathbb{Z}/q_1^{e_1}\mathbb{Z}) \times \cdots \times \widetilde{E}_r(\mathbb{Z}/q_r^{e_r}\mathbb{Z})$, where $\widetilde{E}_i$ is the curve over $\mathbb{Z}/q_i^{e_i}\mathbb{Z}$ defined by:

$$\widetilde{E}_i \colon y^2 = x^3 + ax + b_i \qquad \text{where} \qquad b_i = \left(y_P^2 - x_P^3 - ax_P\right) \bmod q_i^{e_i}.$$

More precisely, consider a prime factor q_i of p' of multiplicity 1 ($e_i = 1$) such that the curve $\widetilde{E}_i$ is indeed an elliptic curve over $\mathbb{F}_{q_i}$ (i.e. q_i is prime to $2 \cdot 3 \cdot (4a^3 + 27b_i^2)$, which happens with high probability unless q_i is very small), and let $P_i \in \widetilde{E}_i(\mathbb{F}_{q_i})$ be the point $\left(x_P \bmod q_i, y_P \bmod q_i\right)$. Then, the double-and-add algorithm carried out in Jacobian coordinates modulo q_i correctly computes the Jacobian coordinates of the scalar multiple $kP_i \in \widetilde{E}_i(\mathbb{F}_{q_i})$, *unless* the validity condition for the addition law[1] described in Section 2 (saying that the points being added are different, not inverses of each other and not at infinity) fails at some point in the computation, in which case it is easy to check that the computation returns either the point at infinity on $\widetilde{E}_i$ or the erroneous value $(0 : 0 : 0)$. The same holds modulo primary factors $q_i^{e_i}$ of p of higher multiplicity, except that zeros are replaced by non-invertible elements of $\mathbb{Z}/q_i^{e_i}\mathbb{Z}$.

As a result, it follows from the Chinese Remainder Theorem that when the whole computation is carried out modulo p', the algorithm returns a value $(X' : Y' : Z')$ which, when reduced modulo $q_i^{e_i}$, gives the correct scalar multiple $kP_i \in \widetilde{E}_i(\mathbb{Z}/q_i^{e_i}\mathbb{Z})$ whenever $\widetilde{E}_i$ is indeed an elliptic curve and q_i does not divide Z'. If, moreover, the order N_i of P_i in the group $\widetilde{E}_i(\mathbb{Z}/q_i^{e_i}\mathbb{Z})$ is not too large (hopefully much smaller than 2^n), we can hope to recover $k \bmod N_i$ in time $O(\sqrt{N_i})$ using a generic algorithm for the discrete logarithm such as

[1] Note that a typical implementation of scalar multiplication on the curve can omit checking whether the validity condition holds, as it is never satisfied under normal conditions, but even if the check is carried out, it is immaterial for our purposes. Indeed, the check would be done using equality modulo p', and hence would not detect equality modulo a single prime factor of p'.

Pollard's rho.[2] And ultimately, putting all those results together, we should obtain $k \bmod \nu$ where ν is the LCM of the N_i's corresponding to those primary factors for which the computation could be carried out.

There are almost always primary factors for which the computation fails, and even for those where it succeeds, the corresponding N_i's may not be relatively prime, so one cannot hope to recover all of k using a single faulty computation. But as we will see, we can actually come quite close! Our analysis of the attack will proceed as follows:

- In Section 3.2, we assume that the faulty modulus p' is obtained from p by flipping a single chosen bit, and we show that if that bit is selected correctly, we can make p' smooth enough that the discrete logarithm becomes easy in all reduced curves $\widetilde{E}_i$.
- In Section 3.3, we then show that the output of a single faulty double-and-add computation in Jacobian coordinates is then enough to recover $n - O(\log^2 n)$ bits of k (out of n), which is quite close to optimal.
- In Section 3.4, we discuss how the few remaining unknown bits of k can be recovered easily if the correct value of $kP \in E(\mathbb{F}_p)$ is known.[3]
- In Section 3.5, we show that recovering k is still possible if the result of the scalar multiplication is converted back from Jacobian to affine coordinates before being output, under plausible assumptions on the Jacobian-to-affine conversion algorithm. Furthermore, we show that the affine x-coordinate alone is enough to carry out the attack.
- In Section 3.6, we discuss how this fault attack on scalar multiplication easily extends to a full key recovery attack on ECDSA signatures given only two faulty signatures.
- Finally, while all the previous results are obtained assuming chosen bit flip faults, we briefly discuss in Section 3.7 how they extend to the case of random faults when a higher number of faulty results is available.

3.2 Choice of the Faulty Modulus

In Ciet and Joye's fault attack [11], of which the present paper is a variant, faults injected on the modulus were considered random. As a result, for a 256-bit elliptic curve, say, the faulty modulus p' would typically have a prime factor of

[2] In practice, N_i is often composite, so it is preferable to first factor this order and then use the Pohlig–Hellman algorithm. Moreover, when $e_i > 1$, we can recover $k \bmod N_i$ even faster by first carrying out the discrete logarithm computation in the reduced curve modulo q_i, and then lifting to q_i^2, q_i^3 and so on until $q_i^{e_i}$. However, factors of p' with multiplicity higher than 1 are usually very small if they exist at all, so it is rarely useful to treat them separately in practice.

[3] Whether the correct value of kP is available to an attacker depends on the protocol implemented by the device. This is typically the case when k is the secret key itself, as in static Diffie–Hellman key exchange, BLS signatures and many other protocols. In other settings like ECDSA, k is randomly chosen anew in each execution of the protocol, but as we show below, we can then break the corresponding schemes using several faults.

size $256\lambda = 256 \cdot 0.642\ldots \approx 164$ bits (where λ is the Golomb–Dickman constant), making the discrete logarithm problem on the corresponding reduced curve likely intractable. As a result, Ciet and Joye's attack can usually only recover a small fraction of the bits of k using a single fault.

By contrast, our analysis is based on the assumption that the attacker can *choose* the fault to some extent: more precisely, we assume that the attack gets to flip a single chosen bit of p. This is a rather powerful fault model, but this type of faults has been shown by Agoyan et al. [1] to be consistently achievable in practice using laser shots on SRAM cells.

In such a setting, we claim that we can ensure that the faulty modulus p' has only relatively small prime factors, making the discrete logarithm probably easy in all reduced curves.

Indeed, there are n integers obtained from p by flipping a single bit, and with respect to the distribution of the sizes of their prime factors, it is reasonable to make the heuristic assumption (as is usually done in the complexity analysis of e.g. factoring algorithms) that they behave like independent random numbers of the same size; our experiments seem to confirm that this assumption holds. Now, recall that for any constant $s > 1$, the asymptotic probability that a random integer x is $x^{1/s}$-smooth (i.e. has all of its prime factors smaller than $x^{1/s}$) is given by Dickman's function $\rho(s)$ [13], which satisfies $\log \rho(s) \sim -s \log s$ (see e.g. [21]).

As a result, the probability that all of the integers obtained from p by flipping a single bit have a prime factor of at least n/s bit is given by $\big(1-\rho(s)\big)^n \approx \exp\big(-n\rho(s)\big)$, which is bounded by $1/e$ as soon as $n\rho(s) \geq 1$, or equivalently, $\log n \geq -\log \rho(s) \sim s \log s$. Therefore, with good probability, at least one of the integers obtained from p by flipping a single bit is $2^{n/s}$-smooth with $s \approx \log n / \log\log n$.

We have seen that our fault model essentially reduces the discrete logarithm problem in $E(\mathbb{F}_p)$ to ECDLP instances in elliptic curves $\widetilde{E}_i$ over base fields $\mathbb{F}_{q_i}$ corresponding to the prime factors[4] q_i of the faulty modulus p'. Since we can choose p' to be $2^{n \log\log n/\log n}$-smooth with good probability by flipping a bit, we obtain that, in this setting, the heuristic complexity of ECDLP-using-a-single-fault is bounded by $O(2^{n \log\log n/2\log n})$ which is subexponential in n.

More practically, if we consider 256-bit elliptic curves, say, we find that $\big(1-\rho(s)\big)^{256}$ is $0.466\ldots < 1/2$ for $s = 4.2$, and $256/4.2 = 60.9\ldots$. Therefore, this fault attack will typically reduce 256-bit ECDLP to discrete logarithm problems in a few random curves over base fields of less than 61 bits, which are quite easy to solve in practice. Note also that effectively finding the correct bit to flip amounts to factoring 256 numbers of 256 bits each, which can be done in matter of minutes on a standard desktop computer.

The next few sections are devoted to making precise, using properly justified heuristics, in what sense the fault does indeed "reduce" the larger ECDLP instance to the smaller ones.

[4] Due to the lifting technique mentioned in a previous note, this is correct even if p' has repeated prime factors, but we reiterate that such factors of higher multiplicity are irrelevant in practice: a prime factor q of p' is repeated with probability $\approx 1/q$, which is small enough to ignore if q is larger than a dozen bits or so.

3.3 Result in Jacobian Coordinates: Recovering Most of the Scalar

In this section, we argue that an attacker who obtains the result of a single faulty scalar multiplication by k as a point in Jacobian coordinates (i.e. that is not converted back to affine coordinates) can recover almost all the bits of k. This is done in three steps. We first show that the computation has a high probability to succeed modulo all the prime factors q_i of p' such that $q_i \gg n \log n$. We call those prime factors "good moduli", and the other ones "bad moduli". We then prove that the bad moduli account for only a small part of the bits of p', in the sense that the bit length of the $O(n \log n)$-smooth part of p' is bounded as $O(\log n)$ with high probability. And finally, we show that, under reasonable heuristics, the bit size of the LCM ν of the orders $N_i = \mathrm{ord}(P_i)$ is not much smaller than n (it is of length $n - O(\log^2 n)$ bits). As a result, since we finally obtain $k \bmod \nu$, our attack recovers $n - O(\log^2 n)$ bits of information out of n on the scalar k using just a single fault.

The Computation Succeeds for Good Moduli. First, we observe that the erroneous computation is unlikely to happen on the large modulus. For a given q_i, we show that the probability that the computation kP on $\widetilde{E}_i(\mathbb{F}_{q_i})$ meets the point at infinity is $O(\frac{n \log q_i}{q_i})$, where n denotes the bit length of k. Our analysis is based on the following reasonable heuristic assumptions:

1. the curve $\widetilde{E}_i$ behaves like a random elliptic curve over $\mathbb{F}_{q_i}$;
2. the point P behaves like a random point in $\widetilde{E}_i(\mathbb{F}_q)$; and
3. the scalar k is a random n-bit integer.

Note that the probability mentioned above is very small for a prime $q_i \gg n \log n$. It is also consistent with the intuition that the point at infinity is much more likely to be encountered as part of the scalar multiplication on a small elliptic curve group than on a large one (and the Hasse–Weil bound implies that the group size is essentially given by the size of the base field).

Assume that the scalar multiplication by k is carried out in the double-and-add approach. The computation fails if a multiple of the order of the point appears somewhere during the scalar multiplication. Under the heuristic assumption that each constant appearing in the scalar multiplication behaves like a random integer of the corresponding bit length, we prove in the full version of this paper [28] that the probabilty of encountering the point at infinity is bounded by $2n/\mathrm{ord}(P)$ for the n-bit scalar multiple k.

Thus for a given curve E defined over $\mathbb{F}_q$, the probability that a random point P would meet the point at infinity becomes $\sum_d \Pr[\mathrm{ord}(P) = d] \cdot (2n/d)$, where d runs over the divisors of the order of the elliptic curve. This leads us to consider the function $\sum_d \frac{\Pr[\mathrm{ord}(P)=d]}{d} = \frac{1}{|E|} \sum_{P \in E} \frac{1}{\mathrm{ord}(P)}$. In the full version of this paper [28], we show that the expected value of $\sum_{P \in E} \frac{1}{\mathrm{ord}(P)}$ asymptotically becomes $O(\log q)$. Consequently, we obtain that the probability $O(\frac{n \log q}{q})$, which is small for $q \gg n \log n$. For example, this explains that for 256-bit elliptic curve the result of the faulty computation gives a correct value on the corresponding curves for large moduli q_i such that $q_i \gg 256 \cdot 8 = 2048$.

Bad Moduli Account for a Small Number of Bits. From the previous subsection, we expect that the computation modulo large factors of p' succeeds with high probability. However, the computation usually fails to give a useful result on "bad moduli" $q_i = O(n \log n)$. Fortunately, in this section we show that the size of the product of these bad moduli is not so large.

For integers B and x, we write $S_B(x)$ for the B-smooth part of x, i.e. the product of all prime factors of x (with multiplicities) which are $\leq B$. Moreover, if u is a B-smooth integer, we define $P_B(u)$ as the asymptotic probability that a random integer x satisfies $S_B(x) = u$. This probability is well-defined, and given by $P_B(\ell^a) = \frac{1}{\ell^a} - \frac{1}{\ell^{a+1}}$ on powers of primes $\ell^a \leq B$, from which the value of $P_B(u)$ for all B-smooth integers u easily follows. Now consider the probability that $S_B(p') \leq B^\alpha$, i.e. $\sum_{u \leq B^\alpha} P_B(u)$. In the full version of this paper [28], we prove:

Theorem 1. *Let $S(x, y)$ be the set of y-smooth integers up to x. For any positive real number $\alpha > 0$, we have*

$$\sum_{u \in S(B^\alpha, B)} P_B(u) \sim \frac{1}{e^\gamma} \cdot \int_0^\alpha \rho(s) ds$$

as $B \to \infty$. Here, ρ is Dickman's function and γ is the Euler-Mascheroni constant.

Consider the right-hand side of the asymptotics of Theorem 1. By definition of Dickman's function, $\rho(s) = \frac{1}{s} \int_{s-1}^{s} \rho(t) dt$ for all $s > 1$ and $\rho(s) = 1$ for $0 \leq s \leq 1$. It follows that $\rho(s) = 1 - \log s$ for $1 \leq s \leq 2$, and a simple calculation then gives:

$$\frac{1}{e^\gamma} \int_0^2 \rho(s) ds = \frac{3 - 2\log 2}{e^\gamma} = 0.90603 \cdots .$$

It means that for random n-bit integer p', the inequality $S_B(p') \leq B^2$ holds with about 91% probability. More generally, for $B = O(n \log n)$, the bit length of $S_B(p')$ is $O(\log n)$ with high probability. And in the case of a 256-bit random integer p', we can expect that the product of primes less than $B \approx 256 \cdot 8 = 2048$ is less than $2 \log 2048 = 15.2 \ldots$ bits with high probability.

Size of the LCM of the Good Orders. For each of the good moduli q_i, the faulty computation reveals the point $k\widetilde{P}_i \in \widetilde{E}_i(\mathbb{F}_{q_i})$, and since the discrete logarithm problem on each of these small curves is tractable, we can solve it to obtain $k \bmod N_i$ where $N_i = \mathrm{ord}_{\widetilde{E}_i}(P)$. Then, the total recoverable bit length of k is determined by the size of $\nu := \mathrm{lcm}\{N_i\}$, where the lcm is taken over the set I of indices of the good moduli q_i. Thus, to show that most bits of k can be recovered, we would like to justify that the size of ν is close to n.

This is done in two steps. First, we argue that for each i, the order N_i of $\widetilde{P}_i$ in $\widetilde{E}_i(\mathbb{F}_{q_i})$ is close to the order of the group itself. This can be done under the heuristic assumption that $\widetilde{P}_i$ is a random point on the corresponding curve, and that the curve itself is random in an appropriate sense. Indeed, we can explicitly

compute the expected order of a random element in a finite abelian group with at most two invariant factors (it is close to the exponent of the group), and the order of the smaller invariant factor of a random elliptic curve is typically small (precise estimates can be found in [20]). Overall, we prove in the full version of this paper [28] that N_i, the order of P on $\tilde{E}_i$, is only a constant number of bits shorter than $|\widetilde{E}_i(\mathbb{F}_{q_i})|$ on average.

As a result, of the LCM ν of the N_i's will be within a constant factor of the LCM of the curve orders $N'_i = |\widetilde{E}_i(\mathbb{F}_{q_i})|$. The second step is then to justify that this latter LCM is close to the product of the N'_i's, or equivalently of the q_i's, which we know is of size $n - O(\log n)$ bits. To do so, we argue, using the results of Gekeler [20] again, that the distribution of the sizes of the prime factors of the order a random elliptic curve is essentially the same as that of a random integer of the same size. Thus, the LCM of the N'_i's should behave like the LCM of $|I|$ random integers of the same sizes. And a recent theorem of Fernández and Fernández [17] provides a bound on the difference between the size of the product and the size of the LCM: it ensures that $\log \prod_{i \in I} N_i - \log \nu = O(|I|^2) = O(\log^2 n)$.

All in all, under reasonable heuristic assumptions, the total recoverable bit length of k is at least $n - O(\log^2 n)$. A more detailed discussion of these heuristic arguments, and full proofs under the appropriate assumptions, are provided in the full version of this paper [28].

3.4 Result in Jacobian Coordinates: Recovering the Whole Scalar

Let $(X_Q : Y_Q : Z_Q)$ be a point in the Jacobian coordinate of the result of the faulty computation with modulus p'. If the scalar multiplication on the modulus q_i contains the point at infinity somewhere, then the Z-coordinate of the result should be zero. Thus, in practice, we detect which modulus is a good modulus by checking the value of the GCD of Z_Q and p'; the good modulus q_i never be a factor of $d := \gcd(Z_Q, p')$.

With the proper choice of the faulty modulus, we have seen how to recover a large part of the exponent, namely $k \pmod{\nu}$ where ν is a known integer of expected bit size $n - O(\log^2 n)$. Assume that we also obtain the value of the *correct* computation kP. From $k = \nu \mathtt{x} + (k \bmod \nu)$ for some $\mathtt{x}$ in a small interval of length $O(\log^2 n)$, we can recover $\mathtt{x}$ by applying Pollard's lambda algorithm to the pair of known points νP and $\mathtt{x} \cdot (\nu P) = kP - (k \bmod \nu)P$. This only requires $O(\sqrt{\log^2 n}) = O(\log n)$ arithmetic operations on the elliptic curve $E(\mathbb{F}_p)$, and is therefore quite fast (and in any case, polynomial time).

3.5 Result in Affine Coordinates

In practice, at the end of the computation, the final result is usually converted back to affine coordinates before being output. So, the final point $(X_Q : Y_Q : Z_Q)$ in the Jacobian coordinate should be converted into the affine coordinate $(X_Q/Z_Q^2, Y_Q/Z_Q^3)$. This step requires the inversion of Z_Q in $\mathbb{F}_p$. To invert the element Z_Q modulo p, it is widely used the extended Euclidean algorithm (EEA):

Find integers α and β satisfying $\alpha Z_Q + \beta p = 1$, then compute $(\alpha^2 X_Q \bmod p, \alpha^3 Y_Q \bmod p)$ from the point $(X_Q : Y_Q : Z_Q)$.

In our fault model the inverse of Z_Q does not exist with respect to the faulty modulus p', because Z_Q and p' are not relatively prime in general. However, if we assume that the procedure of conversion from the Jacobian to the affine coordinates is done by the EEA without checking that $\gcd(Z_Q, p') = 1$ (this is a reasonable assumption in the sense that this check is never necessary for field arithmetic: a field element is invertible as soon as it is non-zero, and this is normally verified separately), we obtain the faulty affine coordinates $(\tilde{x}, \tilde{y}) = (\tilde{\alpha}^2 X_Q \bmod p', \tilde{\alpha}^3 Y_Q \bmod p')$, where $Q = (X_Q : Y_Q : Z_Q)$ is the result of computing kP modulo p' and $\tilde{\alpha} Z_Q + \tilde{\beta} p' = \gcd(Z_Q, p')$ in the EEA.

To obtain the value of $kP_i \in \tilde{E}_i$, we need to compute the correct affine coordinates:

$$(x_i, y_i) = (X_Q/Z_Q^2 \bmod q_i, Y_Q/Z_Q^3 \bmod q_i)$$

from the given result $(\tilde{x}, \tilde{y})$. Note that $\frac{Z_Q}{d}$ and $\frac{p'}{d}$ are relatively prime for $d := \gcd(Z_Q, p')$, and we have $\tilde{\alpha} \cdot \frac{Z_Q}{d} + \tilde{\beta} \cdot \frac{p'}{d} = 1$. This induces that $Z_Q^{-1} = \frac{\tilde{\alpha}}{d} \bmod q_j$ for prime factors q_j of $\frac{p'}{d}$, so we deduce that

$$\tilde{x}/d^2 = X_Q/Z_Q^2 \bmod q_j \quad \text{and} \quad \tilde{y}/d^3 = Y_Q/Z_Q^3 \bmod q_j.$$

It remains to recover d from $\tilde{x}$ and $\tilde{y}$. Note that d is the product of the prime factors q_i on which $Z_Q \bmod q_i = 0$.

If the computation of kP involves the point at infinity somewhere for $P \in E_i(\mathbb{F}_{q_i})$, then the final result is of form $(0 : 0 : 0)$. Thus $\gcd(Z_Q, p')$ divides $\gcd(X_Q, p')$.

Conversely, assume that $Z_Q \neq 0 \bmod q_i$. The point $(X_Q/Z_Q^2, Y_Q/Z_Q^3) \bmod q_i$ behaves as a random point on $E_i : y^2 = x^3 + ax + b_i$. The curve E_i has a point of form $(0, y)$ if and only if b_i is a quadratic residue modulo q_i and in this case we have $(0, y) = (0, \pm b_i^{1/2})$. So, the probability that a randomly chosen point has its x-coordinate zero is $\frac{2}{|E_i|} \approx \frac{2}{q_i}$ if it exists. Thus $X_Q \neq 0 \bmod q_i$ with the probability $1 - \frac{1}{2} \cdot \frac{2}{q_i} = 1 - \frac{1}{q_i}$. Thus $\gcd(X_Q, p')$ divides $\gcd(Z_Q, p')$ with probability $\prod_{i=1}^{\tau}(1 - 1/q_i)$. We deduce that $d = \gcd(Z_Q, p') = \gcd(X_Q, p')$ with high probability.

Finally, we have $\gcd(\tilde{x}, p') = \gcd(\tilde{\alpha}^2 X_Q, p') = \gcd(X_Q, p') = d$, since $\tilde{\alpha}$ is relatively prime to p'/d, and we are able to recover the correct affine coordinates from $(\tilde{x}, \tilde{y})$. The recovery of the exponent can be carried out as before.

Remark 1. Assume that we only have the affine x-coordinate of the faulty computation. In this case, we have two choices of possible point $(x_i \bmod q_i, y_i \bmod q_i)$ for each i. Therefore, if there are w good moduli, we have 2^w possibilities for $k \bmod \nu$. If the exact value of kP is known, we can thus recover k by running Pollard's lambda algorithm 2^w times as before using all of the possible candidates for $k \bmod \nu$. This exhaustive search is quite fast, and in any case polynomial time.

3.6 Attack on ECDSA

Consider two faulty signatures of form

$$(r_1, s_1) := (x_1, {k_1}^{-1}(h_1 + r_1 \cdot \mathtt{d}) \bmod N) \quad \text{for } h_1 = H(m_1)$$

and

$$(r_2, s_2) := (x_2, {k_2}^{-1}(h_2 + r_2 \cdot \mathtt{d}) \bmod N) \quad \text{for } h_2 = H(m_2).$$

As before, we know that $k_1 = \nu \mathtt{x}_1 + \eta_1$ and $k_2 = \nu \mathtt{x}_2 + \eta_2$ for the small unknowns $0 < \mathtt{x}_1, \mathtt{x}_2 < 2^{O(\log^2 n)}$. To find $\mathtt{x}_1$ and $\mathtt{x}_2$, we consider the equations obtained by multiplying k_1 and k_2 for each s_1 and s_2:

$$\begin{cases} s_1 \cdot (\nu \mathtt{x}_1 + \eta_1) = h_1 + r_1 \cdot \mathtt{d} \bmod N \\ s_2 \cdot (\nu \mathtt{x}_2 + \eta_2) = h_2 + r_2 \cdot \mathtt{d} \bmod N. \end{cases}$$

Eliminating $\mathtt{d}$, we obtain the equation

$$r_2 s_1 (\nu \mathtt{x}_1 + \eta_1) - r_1 s_2 (\nu \mathtt{x}_2 + \eta_2) = r_2 h_1 - r_1 h_2 \bmod N.$$

Hence the problem reduces to solving the bivariate modular equation $\alpha \mathtt{x}_1 + \beta \mathtt{x}_2 = \gamma \bmod N$ for small $\mathtt{x}_1$, $\mathtt{x}_2$ and known N which can be solved efficiently with the LLL algorithm when $|\mathtt{x}_1|, |\mathtt{x}_2| \lesssim N^{1/2}$ (since it is an instance of the $(1, 2)$-ME problem in the terminology of Takayasu and Kunihiro [34]; see also [12]), and that bound is of course satisfied in our setting.

3.7 Extending the Attacks to Random Faults

Suppose that we are unable to obtain faults on a specific bit of p, but can instead carry out several faulty executions of the scalar multiplication with the same scalar k in which a different random fault is injected in p every time. Ciet and Joye have shown that, in such a setting, we can assume that each of the faulty moduli p'_j is known, as it can be recovered from the resulting point [11].

Our attack naturally extends to such a setting as follows: while some "good moduli" among the largest factors of the p'_j's may be too large for the corresponding discrete logarithm instances to be tractable, we can use all the results from the available good moduli starting from the easiest Pohlig–Hellman instances, and stop as soon as enough bits have been recovered in the LCM so that all of k is known. Experimentally, we find that for a 256-bit modulus, 3 to 4 random faults are enough to recover all of k in a few minutes. The precise analysis of this multiple random fault variant is left as future work. Note that this attack is similar to the one originally proposed by Ciet and Joye, but does account for "bad moduli" and can therefore be carried out in practice.

Similarly, it is even easier to adapt the ECDSA attack to random faults: if each random fault reveals a bit more than one fourth of the nonce on average, say, the LLL attack above generalizes directly to a full key recovery attack using 4 faulty signatures.

4 Simulation Results

We have implemented the attacks described in the previous section in the Sage computer algebra system [33], and carried out simulations on a mid-range workstation with 16 Xeon E5 CPU cores at 2.9 GHz. All our code was run on a single core of that machine, except the search for optimal bit flips, which was parallelized.

We experimented with several standardized elliptic curve parameters: the 224-bit and 256-bit prime field NIST curves [18], the Brainpool curves of the same sizes [30], and the 256-bit curve recommended by the French government [4]. Indeed, the base fields of NIST curves have a special form, which we thought could yield to a somewhat special behavior with respect to our attacks.

4.1 Search for Optimal Bit Flips

Given an elliptic curve $E/\mathbb{F}_p$ and a corresponding base point $P \in E(\mathbb{F}_p)$, we search for "good" bit faults by trying, for each bit of p, to flip that bit, which yields a certain faulty modulus p'. We then compute the prime factorization $\prod q_i^{e_i}$ of p', and the order N_i of each of the reduced points P_i on the reduced curves $E_i(\mathbb{Z}/q_i^{e_i}\mathbb{Z})$ (for those of them that are actually elliptic curves).

A faulty p' is good if it is quite smooth (i.e. if the bit size of the largest q_i is relatively small), or more precisely if the N_i's themselves are smooth, as the complexity of recovering the discrete logarithm using the Pohlig-Hellman is essentially given by the size of the LCM ν of the N_i's.

It turns out that while the smoothest possible p' is sometimes optimal in the latter respect (this is the case for example for the French government curve FRP256v1, for which flipping bit 184 of the modulus gives both the smoothest p', with a largest prime factor of 59 bits, and the smoothest ν, with a largest prime factor of 35 bits), it is not always the case. For example, although the smoothest possible p' for NIST curve P–256 is obtained by flipping bit 64 (2^{51}-smooth), it is much better to flip bit 5 of NIST curve P256: the corresponding p' is only 2^{113}-smooth, but ν is then 2^{23}-smooth (vs. 2^{34}-smooth for bit 64), providing significantly faster discrete logarithm recovery.

To find the optimal bit flip, we can thus rank each bit position i of p according to the size of the largest prime factor of ν associated with the faulty modulus $p' = p \oplus 2^i$, and keep the best choice. Sometimes, the second or third best choice can be preferred if the size of the largest prime factor of ν is close, but ν itself is larger, as this makes the Pollard lambda step of the attack faster.

By far the most costly part of the attack is the factorization of the faulty moduli p'. Factoring integers of 256 bits or less is not hard on modern computers but can still take a substantial amount of time in unfavorable cases: using the basic `factor` algorithm in Sage (which uses PARI's implementation of ECM and MPQS), we found that the whole search took around 2.5 CPU hours for a 256-bit curve on a single core. Clearly, the search can easily be parallelized by running the computations for different bits i separately. We do it using the `@parallel`

Table 1. Results of the search for optimal bit flips on five standard curve parameters. For each curve, we provide the bit flip yielding the smoothest p', the smoothest ν, and the one we consider optimal for our purposes (ranked 1 to 3 in order of smoothness for ν). The top-left cell "128 (46, 44, 134)" means that flipping bit 128 of the modulus for the curve P–224 yields a 2^{46}-smooth p', and a 2^{44}-smooth ν which is 134-bit long in total. Computation time is on the wall clock, on our 16-core machine.

	NIST		Brainpool		French gov.
Bit flips	P–224	P–256	P224t1	P256t1	FRP256v1
Smoothest p'	128 (46, 44, 134)	64 (51, 34, 227)	58 (51, 26, 213)	24 (60, 48, 228)	184 (59, 35, 218)
Smoothest ν	82 (60, 29, 219)	5 (113, 23, 232)	10 (71, 23, 193)	16 (88, 35, 239)	184 (59, 35, 218)
Recommended	82 (60, 29, 219)	5 (113, 23, 232)	58 (51, 26, 213)	16 (88, 35, 239)	39 (86, 36, 236)
Time (s)	116	822	109	887	854

decorator in Sage, which speeds up the computation by a factor of around 11 on our 16-core machine. Detailed results are provided in Table 1.

4.2 ECDLP Attack

As described earlier, given the result of a faulty scalar multiplication kP carried out with our chosen faulty modulus p', we recover the exponent k modulo the order N_i of P on each of the reduced curves using the Pohlig–Hellman algorithm, and combine those result with the Chinese Remainder Theorem to get k modulo ν. Note that for a given exponent k, the number of bits recovered at this step may be lower than the theoretical maximum computed previously, in case the point at infinity on one of the reduced curves is reached at some stage during the scalar multiplication.

If in addition the result of the correct multiplication kP on the curve E is available, we use Pollard's lambda algorithm to find the remaining bits of k.

Our implementation uses Sage's generic group algorithms for both Pohlig–Hellman and Pollard lambda, which is certainly suboptimal, having both high overhead from the unoptimized Python code and naive underlying elliptic curve arithmetic. Nevertheless, the timings that we obtain and report in Table 2 confirm that the attack is very practical.

4.3 ECDSA Attack

In the ECDSA attack, we are given two faulty ECDSA signatures generated with our chosen faulty modulus. We first use our ECDLP attack to recover more than half of the bits of the nonce used in each signature (we can stop when those bits are obtained: we don't have to go through all possible reduced points). We actually recover a set of candidate nonce values to exhaustively search over, due to the sign indeterminacy discussed in the previous section. We then iterate over this exhaustive search space trying to recover the remainder of the nonces, and hence the full ECDSA secret key, using LLL lattice reduction.

Table 2. Ranges for the number of bits recovered and the recovery time in the main (Pohlig–Hellman) and final (Pollard lambda) steps of the ECDLP attack. Each recovery has been carried out on 5 random exponents for each curve, using the optimal bit flip computed before. Timings are given on a single CPU core of our machine.

	NIST		Brainpool		French gov.
Measured attacks	P–224	P–256	P224t1	P256t1	FRP256v1
Main recovered bit size	205–205	232–232	213–213	239–239	228–236
Main recovery time (s)	6.6–7.8	1.5–1.7	2.4–3.1	52–59	27–34
Remaining recovered bit size	20–20	25–25	11–11	18–18	21–29
Remaining recovery time (s)	0.29–2.4	1.0–7.1	0.08–0.10	0.15–0.59	0.54–8.1

Table 3. Timings for the initial as well as LLL stage of the ECDSA attack. Timings are given on a single CPU core of our machine.

	NIST		Brainpool		French gov.
Measured attacks	P–224	P–256	P224t1	P256t1	FRP256v1
Recovered nonce size	153–153	178–178	134–134	182–182	177–177
Number of nonce pairs to search over	64–64	4–4	64–64	16–64	32–64
Nonce recovery time (s)	12–13	2.7–3.2	3.5–4.3	56–68	48–57
LLL recovery time (μs)	2–11	1–2	10–37	4–12	9–27

The attack is very fast, as shown in Table 3. In particular, the LLL step takes negligible time, even though it requires carrying out the exhaustive search discussed in Remark 1.

5 Conclusion

We have proposed and thoroughly analyzed an extension of the Ciet–Joye fault attack on base fields of elliptic curves, and found that it works very well in practice provided that precise enough faults can be obtained on the modulus. It also extends to a very efficient fault attack on ECDSA, using only two faulty signatures.

The original countermeasure suggested by Ciet and Joye (namely, consistency checking of the modulus using a CRC or other cheap redundancy check) does thwart this attack, but our results underscore the importance of using it in actual implementations. Alternatively, using ECC implementations with dedicated arithmetic for a specific base field, as is sometimes done for NIST curves and almost always for rigid curve parameters such as Curve25519 [5] and other "SafeCurves" [6] (but not curves with random base fields like the Brainpool curves [30]), provides intrinsic protection against the attacks from this paper.

Note on the other hand that verifying the signature after generation (a common fault countermeasure especially in the RSA setting) does not help against our attack at all.

We have only considered prime fields, but essentially the same attack carries over in e.g. the binary field setting. It is probably less relevant to practitioners, however, since implementations of binary elliptic curves often use dedicated base field arithmetic as well.

References

1. Agoyan, M., Dutertre, J.-M., Mirbaha, A.-P., Naccache, D., Ribotta, A.-L., Tria, A.: How to flip a bit? In: IOLTS 2010, pp. 235–239. IEEE (2010)
2. Alkhoraidly, A., Domínguez-Oviedo, A., Hasan, M.A.: Fault attacks on elliptic curve cryptosystems. In: Joye, M., Tunstall, M. (eds.) Fault Analysis in Cryptography. Information Security and Cryptography, pp. 137–155. Springer (2012)
3. ANSI X9.63:2001. Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography. ANSI, Washington DC, USA (2001)
4. ANSSI. Publication d'un paramétrage de courbe elliptique visant des applications de passeport électronique et de l'administration électronique française (November 2011), `http://www.ssi.gouv.fr/fr/anssi/publications/publications-scientifiques/autres-publications/publication-d-un-parametrage-de-courbe-elliptique-visant-des-applications-de.html`
5. Bernstein, D.J.: Curve25519: New Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
6. Bernstein, D.J., Lange, T.: SafeCurves: choosing safe curves for elliptic-curve cryptography (2013), `http://safecurves.cr.yp.to` (accessed December 1, 2013)
7. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
8. Biehl, I., Meyer, B., Müller, V.: Differential fault attacks on elliptic curve cryptosystems. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 131–146. Springer, Heidelberg (2000)
9. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. J. Cryptology 14(2), 101–119 (2001)
10. Brier, É., Naccache, D., Nguyen, P.Q., Tibouchi, M.: Modulus fault attacks against RSA-CRT signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 192–206. Springer, Heidelberg (2011)
11. Ciet, M., Joye, M.: Elliptic curve cryptosystems in the presence of permanent and transient faults. Des. Codes Cryptography 36(1), 33–43 (2005)
12. Coron, J.-S., Naccache, D., Tibouchi, M.: Fault attacks against EMV signatures. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 208–220. Springer, Heidelberg (2010)
13. Dickman, K.: On the frequency of numbers containing prime factors of a certain relative magnitude. Arkiv för Matematik, Astronomi och Fysik 22A(10), 1–14 (1930)
14. Fan, J., Guo, X., Mulder, E.D., Schaumont, P., Preneel, B., Verbauwhede, I.: State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In: HOST 2010, pp. 76–87 (2010)

15. Fan, J., Verbauwhede, I.: An updated survey on secure ECC implementations: Attacks, countermeasures and cost. In: Naccache, D. (ed.) Quisquater Festschrift. LNCS, vol. 6805, pp. 265–282. Springer, Heidelberg (2012)
16. Farashahi, R.R., Joye, M.: Efficient arithmetic on Hessian curves. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 243–260. Springer, Heidelberg (2010)
17. Fernández, J.L., Fernández, P.: On the probability distribution of the gcd and lcm of r-tuples of integers. arXiv (2013), http://arxiv.org/abs/1305.0536
18. FIPS PUB 186-3. Digital Signature Standard (DSS). NIST, USA (2009)
19. Fouque, P.-A., Lercier, R., Réal, D., Valette, F.: Fault attack on elliptic curve Montgomery ladder implementation. In: Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC, pp. 92–98 (2008)
20. Gekeler, E.-U.: The distribution of group structures on elliptic curves over finite prime fields. Documenta Mathematica 11, 119–142 (2006)
21. Granville, A.: Smooth numbers: computational number theory and beyond. Algorithmic Number Theory, MSRI Publications 44, 267–323 (2008)
22. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer (2004)
23. IEEE Std 1363-2000. Standard Specifications for Public-Key Cryptography. IEEE (2000)
24. ISO/IEC 18033-2:2006. Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers. ISO, Geneva, Switzerland (2006)
25. ISO/IEC JTC1 SC17 WG3/TF5. Supplemental Access Control for Machine Readable Travel Documents, version 1.01. ICAO (2010), http://mrtd.icao.int/.
26. Joye, M., Tibouchi, M., Vergnaud, D.: Huff's model for elliptic curves. In: Hanrot, G., Morain, F., Thomé, E. (eds.) ANTS-IX. LNCS, vol. 6197, pp. 234–250. Springer, Heidelberg (2010)
27. Joye, M., Yen, S.-M.: The Montgomery powering ladder. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2003)
28. Kim, T., Tibouchi, M.: Bit-flip faults on elliptic curve base fields, revisited. Cryptology ePrint Archive (2014), Full version of this paper, http://eprint.iacr.org/
29. Koblitz, N.: Elliptic curve cryptosystems. Math. Comp. 48, 203–209 (1987)
30. Lochter, M., Merkle, J.: Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. RFC 5639 (Informational) (March 2010)
31. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
32. National Security Agency. The case for elliptic curve cryptography (2005), http://www.nsa.gov/business/programs/elliptic_curve.shtml
33. Stein, W., et al.: Sage Mathematics Software (Version 5.11). The Sage Development Team (2013), http://www.sagemath.org
34. Takayasu, A., Kunihiro, N.: Better lattice constructions for solving multivariate linear equations modulo unknown divisors. In: Boyd, C., Simpson, L. (eds.) ACISP. LNCS, vol. 7959, pp. 118–135. Springer, Heidelberg (2013)
35. Wang, M., Zhan, T.: Analysis of the fault attack ECDLP over prime field. Journal of Applied Mathematics, 1–11 (2011)