Lattice-based weak curve fault attack on ECDSA

Weiqiong Cao[1,2], Hongsong Shi[1], Hua Chen[2], Wei Wei[1], Jiazhe Chen[1]

[1] China Information Technology Security Evaluation Center. Building 1, yard 8, Shangdi West Road, Haidian District, Beijing 100085, China.
[2] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, South Fourth Street 4#, ZhongGuanCun, Beijing 100190, China.
Email: caoweqion@163.com, hsshi@163.com

Abstract. ECDSA algorithm is usually used in ICT system to achieve communication authenticity. But weakness in various implementations of the algorithm may make its security deviate from theoretical guarantee. This paper proposes a new lattice-based weak curve fault attack on ECDSA. An elliptic curve is weak if the problem of ECDLP in a *subgroup* of the point group $\langle G\rangle$ is computationally solvable in practice, where G is the specified basis point of ECDSA algorithm. Since ECDLP is not required to be computationally practical in the whole group of $\langle G\rangle$, our approach extends the known existing attacks along this line. In detail, the proposed attack assumes a fault injection process can perturb a segment of consecutive bits of the curve parameter a in the Weierstrass equation of ECDSA. An analysis on the density of smooth numbers indicates the faulty value a' parameterized elliptic curve is weak in high probability. Then we show the faulty value a' can be recovered by a dedicated quadratic residue distinguisher, which makes it possible to collect enough side channel information about the nonce used in the ECDSA signature generation process. With the help of these information, we can construct a lattice to recover the private key with lattice basis reduction techniques. Further, we show the same strategy can defeat the nonce masking countermeasure if the random mask is not too long, and makes the commonly employed countermeasures ineffective. To our knowledge, the problem remains untractable to the existing weak curve fault attacks. Thus the proposed approach can find more applications than the existing ones. This is demonstrated by the experimental analysis.

Keywords: ECDSA, Weak Curve, Fault Attack, Lattice Attack

1 Introduction

1.1 Existing work on fault attacks

Elliptic curve digital signature algorithm (ECDSA) has found its extensive use in practice. It is mainly used for data authentication in network communication protocols (e.g., TLS protocol), financial IC cards and various embedded cryptographic devices. Over the last decades, side channel attack (SCA) and

fault attack (FA) have been found exploitable on different implementations of ECDSA. Regarding SCA, obtaining useful side channel information is generally the initial step of various attacks. For example, in [14,23,24], an adversary, when collecting enough side channel information about the nonce or some intermediates, can construct specific instances of shortest vector problem (SVP) or closest vector problem (CVP) in lattice, and then employ the lattice basis reduction methods to recover the signature generation key. In [4], Brumley *et al.* shown how to target wNAF scalar multiplication in OpenSSL, and obtain some leaked bits of nonces by timing attack. In [2,1], flush + reload such like attacks have been used to obtain the leaked bits by employing the flaws of instruction cache and scheduling (of CPU). In addition, power analysis [12] and template attack [10] can also be conducted on ECDSA based on valid power traces.

In this paper, what we are interested on is fault attacks. The structure of FA is quite similar to SCA. Firstly, FA manages to obtain valid side channel information about the nonce in ECDSA, then translates the obtained information into a lattice so as to recover the private key through lattice basis reduction algorithms. The difference of FA from SCA is that, the leakage of side channel information in FA is actively induced by fault injection approaches, such as laser injection, electromagnetic injection or voltage glitch interference and so on. The induced signal can perturb the execution flow of the signature generation algorithm, which makes instruction skipped or some intermediates faulty, then makes the target produces faulty signatures. A dozen of fault attacks on ECDSA have been proposed since very early of this century. Here we only review some of them that are related to lattice analysis.

In PKC 2005 [21], Nacache *et al.* introduced a lattice-based fault attack on DSA. In their approach, if some least significant bits of nonce are set to be 0 by inducing voltage glitch, the private key in DSA can be recovered by solving some instance of CVP in lattice. Schmidt *et al.* in FDTC 2009 [27] introduced a new differential fault model. If a point addition or doubling operation during scalar multiplication can be skipped by fault injection, some bits of the nonce can be obtained by differential analysis. Nguyen *et al.* [25] summarized this kind of fault attacks, and called them lattice-based fault attacks. Cao *et al.* in ICISC 2015 [5] also introduced a random fault model targeting the y-coordinate of intermediate point during the calculation of scalar multiplication, which can tolerate more random faulty bits. These fault attacks can be summarized as two types. The first one assumes fault injection [21,25] is induced directly toward the nonce during signature generation, and can make some bits known or fixed. The other one assumes that fault injection is induced into the calculation of scalar multiplication, then differential distinguisher [27,5] is required to recover the bits of the nonce.

In particular, this paper focuses on weak curve fault attacks, which is somehow different from the above categories. In [17], Kim *et al.* showed faults to the modulus p can also be applied to do FA on ECDSA. The attack assumes fault injection can flip some bit of the modulus p, then obtain a weak curve on which solving elliptic curve discrete logarithm problem (ECDLP) is computa-

tionally practical. The solution reveals some leakage information about nonce k, by which two faulty signatures are enough for the lattice attack to recover the private key. However, the approach requires a strong fault model that only one bit (or a few bits) of p is flipped and the faulty modulus p' is known to the adversary. Moreover, it requires all the prime factors p_i of p' and the orders n_i of subgroups $\mathbb{Z}/p_i^{e_i}$ (where $p' = \prod_{i=1}^{u} p_i^{e_i}$, $p_i < p_j$ for $1 \leq i < j \leq u$ and $e_i \in \mathbb{N}$) to be relatively small, such that the time complexity $O(\sqrt{n_u})$ of solving ECDLP in this case is practical. In addition, in order to mount lattice attack, the product $n'(= \prod_{i=1}^{u} n_i)$ of all the orders n_i should satisfy $n' \geq n^{1/2}$, that is, the bit length of n' should be greater than half of the key length of ECDSA, which restrains the applicability of the fault attack.

1.2 Our approach

In this paper, we propose a lattice-based weak curve fault attack on ECDSA. An elliptic curve is weak if ECDLP in a *subgroup* of the point group $\langle G \rangle$ is computationally solvable in practice, where G is the specified basis point of ECDSA algorithm. (See Definition 3 for detail.) Note the definition does not require ECDLP in the whole group of $\langle G \rangle$ being computationally solvable, which is the main difference of our approach from the existing known attacks.

In more detail, we consider a continuous segment of the curve parameter a can be randomly disturbed by fault injection. The faulty value of a, called a' hereafter, is not required to be known, but can be guessed by a specific quadratic residual distinguisher, see the Algorithm ALG-GUESS-PARA in Section 3.2. Then if the induced curve is weak such that ECDLP in some subgroup of $\langle G \rangle$ can be solved practically, we can collect enough reduced information about the nonce. Formally, the weakness is characterized by a factor d of the order n' of G in the weak curve. And the reduced information is expressed in the form of modulo d, see the Algorithm ALG-OBTAIN-NONCEINFO in Section 3.2. Then this type of information can be employed to construct a lattice for key recovery (see Section 3.3). The dimension N of the lattice is tightly related to the module d. In short, the bigger d is, the smaller N would be.

In addition, for ECDSA with random scalar masking, the proposed approach is still practical without any additional masked bits leakage. For example, if $k' = k + \lambda n$, where k is the real scalar and λ is a 64-bit random number, the approach can succeed by selecting bigger modulus d (see Section 3.4).

Our study on the density of smooth numbers shows the probability that n' includes some big prime factors is much greater than that of all its prime factors being small(see Section 3.5). It thus indicates the proposed approach can find more applications than the existing weak curve fault attacks. This is demonstrated by the experimental analysis in Section 4.

2 Preliminaries

In this paper, we consider elliptic curves on prime field $\mathbb{F}_p$, where p is an odd prime.

2.1 Elliptic curve in $\mathbb{F}_p$

Generally, the Weierstrass equation of elliptic curves in $\mathbb{F}_p$ is given by

$$E(a,b): y^2 = x^3 + ax + b \mod p,$$

where parameters $a, b \in \mathbb{F}_p$ satisfy $4a^3 + 27b^2 \neq 0$.

The group of rational points in elliptic curve $E(a,b)$ is defined by

$$\mathbf{G} = \left\{(x,y)|y^2 = x^3 + ax + b \bmod p, x, y, a, b \in \mathbb{F}_p\right\} \cup \{\mathcal{O}\},$$

where $\mathcal{O}$ is the infinite point.

Let G be an element in $\mathbf{G}$ with order n (which is usually a prime), $\langle G\rangle$ be the additive subgroup of $\mathbf{G}$ generated by G. If $P = (x,y) \in \langle G\rangle$, then the inverse element $-P \in \langle G\rangle$ of P is $(x,-y)$. For any integer $k \in \mathbb{Z}_n$, the calculation of $kG = G + G + \ldots + G$ (k times) is called the scalar multiplication in $E(a,b)$, and can be calculated using point doubling and addition operations.

Point Addition

If $P = (x_1, y_1) \in \langle G\rangle$, $Q = (x_2, y_2) \in \langle G\rangle$, and $P \neq \pm Q$, then $(x_3, y_3) = P + Q$ satisfies

$$\begin{aligned} x_3 &= \lambda^2 - x_2 - x_1 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}, \text{ where } \lambda = \frac{y_2 - y_1}{x_2 - x_1}.$$

Point doubling

If $P = (x_1, y_1) \in \langle G\rangle$ and $P \neq -P$, then $(x_3, y_3) = 2P$ satisfies

$$\begin{aligned} x_3 &= \lambda^2 - 2x_1 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}, \text{ where } \lambda = \frac{3x_1^2 + a}{2y_1}.$$

An important notice is that the parameter b is not involved in the calculation of point doubling and addition. The order of $E(a,b)$, denoted by $\#E(a,b)$ can be calculated using SEA algorithm [11].

2.2 ECDSA digital signature algorithm

The ECDSA signature algorithm is described in Algorithm 1 with some less important details being abstracted away.

As shown in Algorithm 1, the randomly generated nonce k is involved in the calculation of ECSM kG (step 3) and the calculation of s (step 6), which are exactly the targets of our attacks.

Algorithm 1 Signature generation of ECDSA

Input: The definition of a specific elliptic curve $\mathrm{E}(a, b)$, a base point G of the curve with order n, private key $d_A \in \mathbb{Z}_n$, message m.

Output: Signature pair (r, s).

1: $e = H(m)$, where H is a cryptographic hash function;
2: Generate k randomly from $\mathbb{Z}_n$;
3: $Q(x_1, y_1) = kG$;
4: $r = x_1 \bmod n$;
5: **if** $r = 0$ **then** goto step 2;
6: $s = k^{-1}(e + d_A r) \bmod n$;
7: **if** $s = 0$ **then** goto step 2;
8: **return** (r, s)

2.3 Smoothness of weak curve order

The following definitions are required to better describe our approach. For all of them, let n be the order of point G in $E(a, b)$.

Definition 1. *Denote the prime factorization of n by $n = \prod_{i=1}^{u} q_i^{e_i}$, where $q_i \in \mathbb{N}$ is a prime factor of n, $e_i > 0$ denotes the degree of q_i in the factorization and $q_i < q_j$ for $1 \le i < j \le u$. For $y \in \mathbb{N}$, if the biggest prime factor q_u meets $q_u \le y$, then n is called y-smooth.*

Definition 2. *The elliptic curve discrete logarithm problem (ECDLP) in $E(a, b)$ is defined as: given $G \in \boldsymbol{G}$ with order n and an element $Q \in \langle G \rangle$, compute the value $k \in \mathbb{Z}_n$ such that $Q = kG$.*

To our knowledge, the best known generic algorithm [31,26] in classical computer for solving ECDLP in arbitrary elliptic curves needs $O(\sqrt{q_u})$ group operations in computation. We call an ECDLP instance practically solvable if its solving complexity is not bigger than a predefined constant PRAC_COMP. In this paper, we set PRAC_COMP$= 2^{64}$ group operations by considering currently achievable computing power of classical computers, which can be redefined to adapt to the future development of computing technology.

Definition 3. *We call n practically* solvable smooth *(with respect to the group $\langle G \rangle$) if the ECDLP on $\langle G \rangle$ is practically solvable. We call n* partially solvable smooth *(with respect to the group $\langle G \rangle$) if there exists a factor d of n such that the ECDLP on $\langle (n/d)G \rangle$ (with order d) is practically solvable. Finally, we call n practically* unsolvable smooth *(with respect to the group $\langle G \rangle$) if n does not belong to the above two cases.*

In this paper, $E(a, b)$ is called a weak (elliptic) curve if the order n of the chosen base point G of $E(a, b)$ is partially solvable smooth or practically solvable smooth.

2.4 Existing fault attacks on weak curves

In this section, we introduce an existing fault attack on weak curves with solvable smooth order [17,3], which can also be used to weak curves with partially solvable smooth order.

It is assumed that the y-coordinate of G is disturbed by a fault injection process, i.e., $G = (x_G, y_G)$ is changed into $G' = (x_G, y_{G'})$ with $y_G \neq y_{G'}$. Then with overwhelming probability, the faulty G' is not on the original curve $E(a, b)$ (the only exception being $y_G = -y_{G'}$). Note since parameter b is not involved in the calculation of scalar multiplication, G' can be viewed on a new curve $E(a, b')$, and then $Q' = kG'$ is calculated on the curve $E(a, b')$, where $b' = y_{Q'}^2 - {x_{Q'}}^3 - ax_{Q'} = y_{G'}^2 - {x_{G'}}^3 - ax_{G'}$.

Assume the induced curve has a solvable smooth order $n' = \prod_{i=1}^{u} q_i^{e_i}$ with respect to the group $\langle G' \rangle$. Then given Q', G' and n', the following approach can be used to compute the scalar k. Firstly, the reduced value $k \bmod q_i$ can be obtained by solving the ECDLP instance $\frac{n'}{q_i}Q' = k\frac{n'}{q_i}G'(i = 1, ..., u)$ with Pollard-rho algorithm [31]. Next, the reduced value $k_i = k \bmod q_i^{e_i}(i = 1, ..., u)$ can be obtained by Pohlig-Hellman algorithm [26]. Finally, the modulo-n' reduced value $t = k \bmod n'$ can be obtained by Chinese remainder theorem(CRT). Hence, $k = t + \mu n'$, where $\mu \in \{0, ..., \lfloor n/n' \rfloor\}$. Enumerate all the possible values of μ to calculate the corresponding k, when $Q = kG$, k is just the correct one that we are looking for.

The above approach shows that n' must be solvable smooth so as to solve ECDLP instances on $\langle G' \rangle$ and $\lfloor n/n' \rfloor \leq$ PRAC_COMP. Otherwise, the approach cannot be applied in practice.

2.5 Lattice basis reduction

Lattice analysis is a key technique to our approach. We thus give some fundamental background about lattice attacks.

Let $B = \{\boldsymbol{b_1}, \ldots, \boldsymbol{b_N}\} \subseteq \mathbb{R}^m$ be a series of N linearly independent vectors. The lattice generated by B is defined as $\mathcal{L}(B) = \left\{\sum_{i=1}^{N} x_i \boldsymbol{b_i} : x_i \in \mathbb{Z}\right\}$, where B serves as a basis for the lattice $\mathcal{L}(B)$, and we call the integers N and m its rank and dimension respectively. If $m = N$, $\mathcal{L}$ is called a full rank lattice with dimension N. The shortest vector problem (SVP) and closest vector problem (CVP) are two computational complexity problems crucial to lattice-based cryptography. We give them below.

Definition 4. *[20]*

(1) **Shortest Vector Problem***: Given a basis of a lattice $\mathcal{L}$, find a lattice vector $\boldsymbol{v} \neq \boldsymbol{0}$ such that $\|\boldsymbol{v}\| \leq \|\boldsymbol{u}\|$ for any nonzero vector $\boldsymbol{u} \in \mathcal{L}$.*
(2) **Closest Vector Problem***: Given a basis of a lattice $\mathcal{L}$ and a target vector $\boldsymbol{t} \in \mathbb{R}^m$, find a lattice vector $\boldsymbol{v} \in \mathcal{L}$ closest to the target $\boldsymbol{t}$, which means $\mathsf{dist}(\boldsymbol{v}, \boldsymbol{t}) \leq \mathsf{dist}(\boldsymbol{u}, \boldsymbol{t})$ for any vector $\boldsymbol{u} \in \mathcal{L}$, where $\mathsf{dist}(\cdot, \cdot)$ denotes the Euclid norm of two points.*

For an N-dimensional approximate SVP, there exist some polynomial-time basis reduction algorithms which can output short lattice vectors when the approximate factor is large enough. Among those algorithms, LLL algorithm [18] is the most typical one, and BKZ-algorithm [7] has been the most practical algorithm for lattice basis reduction based on a series of optimizing technique [28,29].

For random lattices with dimension N, Gaussian heuristic gives a probable estimation on the length of shortest lattice vector in the sense of average as in [22], from which Gaussian expected shortest length of an N-dimensional lattice $\mathcal{L}$ could be defined to be

$$\sigma(\mathcal{L}) = \sqrt{\frac{N}{2\pi e}}\mathsf{vol}(\mathcal{L})^{1/N}.$$

Generally, the actual shortest lattice vector is much easier to be found as the increment of the gap between the shortest length and the Gaussian heuristic. If it is significantly shorter than $\sigma(\mathcal{L})$, it can be located in polynomial time by using LLL and related algorithms. Heuristically, assuming the lattice $\mathcal{L}$ behaves like random, if there exists a lattice vector whose distance from the target is much shorter than $\sigma(\mathcal{L})$, this lattice vector is expected to be the closest vector to the target. Accordingly, this special CVP instance usually could be solved by Babai algorithm or embedding-based SVP.

3 Lattice-based weak curve attack

In this section, we present our lattice-based weak curve attack on ECDSA. The attack consists of two steps: 1) Obtain reduced information of the nonce by weak curve fault attack; 2) Construct an instance of CVP by virtue of the reduced information, and resolve it to recover the private key.

3.1 Fault model

The fault attacks we consider in this paper aim at modifying the curve parameter a by inducing fault to the corresponding physical storage cells (RAM, EEPROM or CPU register for example). Further, we mainly consider a type of random fault, in which a continuous l-bit segment of a is modified randomly by fault injection and the starting bit location of fault is also randomly picked. The fault can be permanent or transient. A permanent fault means the value corresponding to the parameter is definitely changed, and fixed on the faulty value. A transient fault means the parameter keeps the faulty value unless the original value is explicitly restored. The faulty bits length l is usually valued from {1, 8, 16, 24, 32} considering the byte-based cell structure of storage.

Assume, before running the signature generation (Algorithm 1), an adversary induces a permanent (or transient) fault to the parameter a of elliptic curve $E(a, b)$. Denote the modified parameter by a' and suppose it is different from a in a continuous l-bit segment. Therefore, the base point $G = (x_G, y_G)$ will be on a new curve $E(a', b') : y^2 = x^3 + a'x + b' \bmod p$, and $b' = {y_G}^2 - {x_G}^3 - a'x_G \bmod p$.

Correspondingly, the ECSM $Q = kG$ of step 3 in Algorithm 1 will be actually computed on the new curve $E(a', b')$. Let the new order of G be n'. Finally, the faulty signatures (r', s') are output to the adversary.

Note that our fault model has the following limitations: 1) parameter a must be involved in the calculation of ECSM (except that a is sometimes substituted with $p-3$ when $a = p-3$ for sake of resources optimization); 2) There is no point verification checking whether the input point is on the original elliptic curve during the calculation of ECSM. Otherwise, our attack will not work.

3.2 Proposed fault attack on weak curves

Suppose signatures on the weak curve can be retrieved after the signature generation procedure. We will run the following two algorithms sequentially to obtain reduced information about the nonce k. Then, in Section 3.3 we will use the obtained information to construct lattices and recover the private key.

Algorithm ALG-GUESS-PARA**: Guess and determine** a' **and** $x_{Q'}$

Step 1-1. Let $\mathbf{T}$ be a set for storing the possible values of a'. Regarding its initial size, if assuming the fault injection step induced a randomly located and randomly valued continuous l-bit segment errors to a, then initially the number of possible values for a' is $|\mathbf{T}| = (l_a - l + 1)2^l$, where l_a be the bit length of a.

Step 1-2. Run the signature generation procedure to obtain a faulty signature pair (r', s'). Then deduce the x-coordinate $x_{Q'} \bmod p$ of the faulty point Q' from r'. We separate different cases to consider the deduction.

- If $p < n$ then $x_{Q'} \bmod p = r' \bmod n$;
- If $p > n$ and $x_{Q'} \bmod p < n$, then we also have $x_{Q'} \bmod p = r' \bmod n$;
- Regarding the case of $p > n$ but $x_{Q'} \bmod p > n$, we have the deduction as follows. Note $n | \#E(a, b)$ (where $\#E(a, b)$ is the number of point in $E(a, b)$), and in a standard curve the factor h (satisfying $hn = \#E(a, b)$) is usually set to be $h = 1$ or 2. By Hasse theorem [13], we have $p + 1 - 2\sqrt{p} \leq hn \leq p + 1 + 2\sqrt{p}$. Hence, $x_{Q'} \bmod p = r' + \lambda n < p$, and the integer λ can be valued only 1 or 2 when $p > 2^6$, depending on the concrete values of p and n.

Step 1-3. Sieve $\mathbf{T}$ to find valid a'. For each possible $a' \in \mathbf{T}$, calculate $b' = y_G^2 - x_G^3 - a' x_G \mod p$, and for each value $x_{Q'}$ derived in step 2, compute

$$Y = x_{Q'}^3 + a' x_{Q'} + b' \mod p.$$

If Y is a quadratic residue modulo p, keep a' in $\mathbf{T}$; otherwise, eliminate it from $\mathbf{T}$.

Step 1-4. If the size of $\mathbf{T}$ is greater than 1, go to **Step 1-2**; otherwise, regard the only value in $\mathbf{T}$ as the the faulty parameter a'. Then run the SEA algorithm [11] to compute the order n' of G on the curve $E(a', b')$, and factorize it using some subexponential-time algorithms (such as Pollard $p-1$

or number field sieve algorithms). Note when $n' \leq 512$, the factorization step is practical. If the factorization result shows that n' is partially solvable smooth, we get a valid a'; otherwise repeat Algorithm ALG-GUESS-PARA to induce a new curve until getting a valid a' with partially solvable smooth n'. (Note we don't need n' to be *solvable smooth*.) The condition can be satisfied after a number of trials considering the density of smooth numbers (See 3.5 for detail). So in the following we assume there exists a factor d of n' such that d is practically solvable smooth with respect to $E(a', b')$ (see the definitions in Section 2.3). Finally, we end the algorithm.

The above "quadratic residue" distinguisher can eliminate about a half of the invalid values in each invocation. Hence, the total computation complexity is about $O((l_a - l + 1)2^{l+1})$.

Algorithm ALG-OBTAIN-NONCEINFO**: Obtain reduced value of nonce** k

Based the derived valid a', run the following steps to collect as much as useful reduced information about the nonce.

Step 2-1. Run the signature generation procedure to get a signature (r', s'), and based on the derived (a', b') compute possible values $\{x_{Q'}\}$ and Y as above. Since there is only one correct value for $x_{Q'}$, to remove the erroneous computed values, we discard the signature (r', s') if more than one Y derived from it are quadratic residue modulo p, and re-generate a new signature (r', s') until the condition is satisfied. When the correct $x_{Q'}$ on curve $E(a', b')$ is calculated, we obtain two possible points $(x_{Q'}, \pm\sqrt{Y})$ of Q'.

Step 2-2. Without loss of generality, assume $Q'_1 = \left(x_{Q'}, \sqrt{Y}\right) = uG$. Let $\ell = n'/d$, we can solve the problem of ECDLP

$$\ell Q'_1 = u(\ell G)$$

in the d-ordered subgroup $\langle \ell G \rangle$ to obtain $u \bmod d$. Specifically, if Q'_1 is the correct choice of Q', we have $k = u \bmod d$; otherwise if $-Q'_1$ is the correct choice of Q', we obtain $k = (d - u) \bmod d$.

Step 2-3. Repeat the above two steps N times to obtain reduced information about the nonces $\{k_i\}_{i=1}^{N}$, where each reduced information is denoted by

$$k_i = c_i + \lambda_i d$$

for $i = 1, ..., N$, where $c_i = \begin{cases} u_i, \text{ for } y_{Q'} = \sqrt{Y} \\ d - u_i, \text{ for } y_{Q'} = -\sqrt{Y} \end{cases}$ and $0 < \lambda_i < n/d$.

3.3 Proposed lattice-based ECDSA key recovery algorithm

We show how to use the retrieved information about k_i to construct lattices and then recover the private key d_A.

For each $i \in \{1, ..., N\}$, we first assume the correct value of c_i is identified, then we have

$$s_i(c_i + \lambda_i d) = e_i + r_i d_A, \tag{1}$$

where e_i is the hash value of message m_i and $0 < \lambda_i < n/d$. The identification of correct value of c_i is discussed at the end of this subsection.

The equation (1) can be transformed as

$$\lambda_i = {s_i}^{-1} d^{-1} r_i d_A - (d^{-1} c_i - {s_i}^{-1} d^{-1} e_i) \mod n. \tag{2}$$

Let $A_i = {s_i}^{-1} d^{-1} r_i \mod n$, $B_i = d^{-1}\left(c_i - {s_i}^{-1} e_i\right) + n/(2d)$, then there exists a $h_i \in \mathbb{Z}$ such that

$$|A_i d_A + h_i n - B_i| < n/(2d) \qquad (i = 1, ..., N). \tag{3}$$

We can construct a lattice $\mathcal{L}$ by the above inequations (3), and the row vectors $\{\boldsymbol{b}_1, ..., \boldsymbol{b}_{N+1}\}$ of the matrix

$$\mathbf{M} = \begin{bmatrix} n & 0 & \cdots & 0 \\ 0 & n & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & n & 0 \\ A_1 & \cdots & A_N & 1/(2d) \end{bmatrix}$$

construct a basis of $\mathcal{L}$.

Let the target vector $\boldsymbol{t} = (B_1, \ldots, B_N, 0) \in \mathbb{Z}^{N+1}$. There exists a lattice vector $\boldsymbol{v} = \boldsymbol{x}\mathbf{M}$ with the coordinate vector $\boldsymbol{x} = (h_1, \ldots, h_N, d_A) \in \mathbb{Z}^{N+1}$. From inequations (3), we have

$$\|\boldsymbol{v} - \boldsymbol{t}\| < \sqrt{N+1}\, n/(2d). \tag{4}$$

Heuristically, we assume $\mathcal{L}$ is a random lattice. As introduced in Section 2.5, if $\|\boldsymbol{v} - \boldsymbol{t}\|$ is much less than $\sigma(\mathcal{L})$ $(= \sqrt{\frac{N+1}{2\pi e}} \mathrm{vol}(\mathcal{L})^{\frac{1}{N+1}})$, we expect $\boldsymbol{v}$ to be the closest vector to $\boldsymbol{t}$ in $\mathcal{L}$, where $\mathrm{vol}(\mathcal{L}) = \det(\mathbf{M}) = n^N/(2d)$. Hence, it is required

$$\|\boldsymbol{v} - \boldsymbol{t}\| < \sqrt{N+1}\, n/(2d) \ll \sqrt{\frac{N+1}{2\pi e}} \left(n^N/(2d)\right)^{\frac{1}{N+1}}. \tag{5}$$

Let $f = \lceil \log n \rceil$ and $l_d = \lceil \log d \rceil$. If $N > \frac{f + \log\sqrt{2\pi e}}{l_d + 1 - \log\sqrt{2\pi e}}$ and $l_d > \log\sqrt{2\pi e} - 1$, heuristically the inequality (4) can be viewed as a special instance of CVP in lattice $\mathcal{L}$. Consequently, the vector $\boldsymbol{v}$ can be determined by solving the instance of CVP to reveal the private key d_A.

In addition, the inequality (3) is equivalent to

$$|A_i d_A + h_i n - B_i| < n/2^{l_d} (i = 1, ..., N), \tag{6}$$

which is a hidden number problem(HNP)[23,24]. By the same way, it can be transformed into a CVP to recover d_A.

It is assumed above that all the values of $\{c_i\}_{i=1}^{N}$ are correctly guessed before the lattice attack. The reality is that there are two solutions for each c_i after the Algorithm ALG-OBTAIN-NONCEINFO, and it is not sure which one is the correct. Though it is computationally difficult to make it certain in general, but from the above analysis, we know the needed number N of faulty signatures for lattice attack depends on the size of factor d. Or in order to ensure the practicality of the lattice attack, d should not be too small. In fact, the larger d is, the smaller N would be. For example, d is generally recommended to satisfy $d \geq 2^8$ such that $N \approx 45$ for 256-bit ECDSA. So when N is not too big, it is still possible to enumerate all the possible c_i in practice. Specifically, the worse-case time complexity for the lattice attack in this case is $O(2^N T)$, where T represents the time required for each running of the lattice attack.

3.4 Attack on ECDSA with scalar masking

Generally, scalar masking is one of the most common countermeasures for ECDSA to resist SCA. For example, nonce k_i during signature generation is masked as k_i' with a random number α_i, i.e., $k_i' = k_i + \alpha_i n (i = 1, ..., N)$. This countermeasure also could block the existing lattice attacks [14,23,24], since it is required to obtain all the bit leakage information of $\{\alpha_i\}_{i=1}^{N}$. By comparison, our attack is affected much less, specifically as follows.

With the masked nonce k_i', we have

$$Q = k_i' G \text{ and } s_i = {k_i}^{-1}(e_i + r_i d_A) = {k_i'}^{-1}(e_i + r_i d_A) \mod n.$$

Accordingly, the reduced information derived by weak curve fault attack meets $k_i' = c_i + \lambda_i d$, where l_{α_i} denotes the bit length of α_i and $\lambda_i < 2^{f + l_{\alpha_i} - l_d}$. Substitute c_i into s_i and mount lattice attack. If $l_d > \log\sqrt{2\pi e} + l_{\alpha_i} - 1$ and $N > \frac{f + \log\sqrt{2\pi e}}{l_d - l_{\alpha_i} + 1 - \log\sqrt{2\pi e}}$, the private key d_A can be recovered by constructing an instance of CVP. There is no bit leakage of α_i required in our lattice attack except a bigger d. Moreover, the needed d can be obtained with high proportion in experiments. For example, if $l_{\alpha_i} = 32$, the experimental success rate of fault injection is still up to 80% since l_d is recommended as 40 (see Section 4). Obviously, our lattice attack is more practical on ECDSA with scalar masking.

3.5 The density of smooth numbers

When comparing with existing weak curve fault attacks in [17,3,8] (see Section 2.4), our attack puts weaker condition on the process of fault injection. Specifically, the order of the induced weak curve in the proposed method is only required to be partially solvable smooth in the proposed attack, which is weaker than the existing attacks. The following analysis on the smoothness of a random number demonstrates that the weaker condition improves the applicability of proposed attack significantly.

Let z be an integer with prime factorization $z = \prod_{i=1}^{u} p_i^{e_i}$. We say z is y-smooth if $\max_{1 \le i \le u} \{p_i\} \le y$, as mentioned in Section 2.3. We denote by $\psi(x, y)$ the number of integers $z \le x$ such that z is y-smooth. In [9], a result on the bound of $\psi(x, y)$ shows smooth numbers with suitable x, y are relatively common to meet. Specifically, let ϵ be an arbitrary positive constant, then for $x \ge 10$ and $y \ge (\ln x)^{1+\epsilon}$, we have

$$\psi(x, y)/x = e^{-(1+o(1))u \ln u} \qquad \text{as } x \to \infty,$$

where $u = \ln x / \ln y$ and e is the natural number. Note for a fixed x the density of smooth numbers (i.e., $\psi(x, y)/x$) is an increasing function with respect to the bound y of factors. For instance, we can roughly get $\psi(2^{256}, 2^{247})/2^{256} = 0.963$ and $\psi(2^{256}, 2^{128})/2^{256} = 0.25$ (where $o(1)$ is set to be 0 in the approximation). It means smooth (integer) numbers in the scope of $[1, x]$ with at least one large factor could be much more frequent than those with only smaller factors. Note the action of n' is not uniformly random, since the injected fault is only considered to impact very limited bits of the parameter a of the curve. However, as in [9], we make a heuristic assumption that the probability of sampling n' in this method is subject to the density given by $\psi(x, y)/x$. If considering practicality the 2^{128}-smooth n' (with probability 25%) is required by existing weak curve attacks. Our attack extends the possibility of fault attacks on ECDSA with 256-bit private key. In more detail, considering the case that n' is 2^{247}-smooth (with probability 96.3%), though solving ECDLP in the case is currently unpractical, if collecting enough reduced information about the nonce with respect to a small factor d of n' (which is close to 2^9), our attack is still practical. This is supported by the experiments in Section 4.

In conclusion, since with high probability the order n' has at least a big prime factor in factorization (determined by PRAC_COMP), the existing weak curve attacks (which require n' to be practically solvable smooth) may not be efficient in practice. In comparison, our attack can survive in this case with high probability, since we only need n' to be partially solvable smooth. The price is that our attack need more faulty signatures (in number N) to construct lattice, which is much greater than the one in [17]. But this is affordable in practice, since we can usually ask the target to generate enough faulty signatures once fault has been induced, especially when it is permanent.

4 Experimental analysis

In this section, we do the experiments to validate the applicability of the proposed attack. The emphasis is on checking Algorithm ALG-OBTAIN-NONCEINFO and the lattice based key recovery algorithm. Therefore, the fault injection process is not conducted in the experiments. If learning more engineering aspects of fault injection, see the reference [16].

The experiments are conducted in a computer with 3.4GHz 8-core CPU, 8G memory and Windows7 OS. The weak curve order n' (derived by the faulty a')

is calculated by SEA algorithm implemented in Miracl library [19], and the constructed CVP instances are solved by employing BKZ algorithm [7] implemented in NTL library [30] with block size 10.

Two types of 256-bit curve over prime field $\mathbb{F}_p$ are targeted in the experiments, which are NIST P-256 [6] (hereafter called P-256) and the curve recommended in SM2 digital signature algorithm (hereafter called SM2-curve which still can be employed as the curve of ECDSA) [15] respectively. Then for each curve, two types of bit perturbation experiments are simulated, including the single-bit flipped fault and 16-bit random fault. The single-bit flipped fault is to flip a bit-by-bit. Then there are 256 cases in total. The 16-bit random fault is to randomly pick a starting bit location, characterized by a random integer $b \in [0, 240]$, and then XOR the continuous 16-bit segment (identified by b) of parameter a with a 16-bit random number $\beta \in \{0,1\}^{16}$, such that $a' = a \oplus (\beta 2^b)$. The experiments are also done for 256 times. As a whole, four types of different experiments are conducted, and each is done in 256 times.

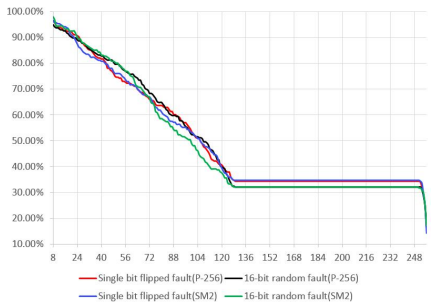

Fig. 1. The proportion γ of partially solvable smooth n' when $l_d > X$

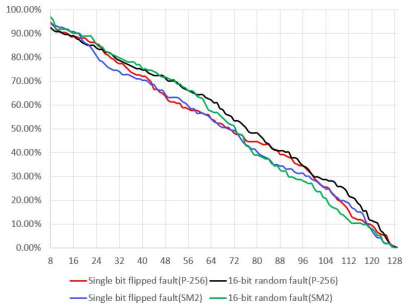

Fig. 2. The proportion γ of unsolvable smooth n' available for our attack when $l_d > X$

We then use the obtained a' to compute n' for each experiment. Figure 1 shows the proportion γ (Y-axis) of partially solvable smooth n' when its factors d satisfies $l_d \geq X$(X-axis), and Figure 2 shows the proportion γ (Y-axis) of the unsolvable smooth n' available for our attack when d satisfies $l_d \geq X$(X-axis) in each type of experiment. (Note each experiment type includes 256 concrete experiments.) When $l_d \geq 8$, there are partially solvable smooth orders with proportion 94.9% at least, which is far greater than the proportion 35% of solvable smooth orders required in the previous weak curve attacks. Moreover, even under the case that n' is unsolvable smooth (with about 65% proportion), there still is 92.5% of unsolvable smooth n' available for our attack. Obviously, most of the weak curves derived from 256 experiments can be applied to our attack, which increases the success rate of fault injection sharply. In addition, no matter whether the curve is P-256 or SM2-curve, and the fault type is single-bit flipped fault or 16-bit random fault, all the proportions of the four types of experiments

are roughly similar with the density mentioned in Section 3.1. Hence, our attack is effective for most of ECC signatures based on prime field.

Finally, based on some faulty a' in SM2-curve, select d with different bit length, and carry on the corresponding lattice attacks. As shown in Table 1, N is the number of signatures required to achieve 100% success rate of lattice attack, T is the time of each lattice attack and O is the maximum complexity of the attack including the enumerating. From Table 1, the complexity of enumerating the correct c_i in lattice attack is computationally feasible even under the worst case $l_d = 8$. In addition, to speed up enumerating, the case $l_d \geq 16$ with time complexity $2^{19}T$ is generally selected in experiments, whose proportion of partially solvable smooth n' is also up to 92.6% at least (See Figure 1). The results show that the success rate of fault injection is significantly high when ensuring the successful lattice attacks.

Table 1. The number of faulty signatures and complexity for lattice attack

Items	$l_d = 8$	$l_d = 9$	$l_d = 16$	$l_d = 32$	$l_d = 64$
N	45	40	19	9	5
$T(s)$	≈ 5.788	≈ 3.675	≈ 0.255	≈ 0.021	≈ 0.005
O	$2^{45}T$	$2^{40}T$	$2^{19}T$	$2^{9}T$	$2^{5}T$

5 Conclusion

We propose a new lattice-based weak curve fault attack on ECDSA which combines the advantages of weak curve fault attack and lattice attack. The order n' of the weak curve generated by faulty a' is not required to be solvable smooth, and the reduced information of nonces is obtained by solving the ECDLP constructed in a small subgroup, by which a new model of lattice attack is constructed to recover the private key. For the single-bit flipped fault or 16-bit random fault, the experiments show the success rate of fault injection that there exists a solvable smooth factor d of n' satisfying $l_d \geq 8$ can be as high as 94.9%. In addition, for ECDSA with w-bits scalar masking, our attack still work with high success rate of fault injection by selecting an appropriate d satisfying $l_d - w > \log\sqrt{2\pi e} - 1$.

References

1. D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom. Ladderleak: Breaking ECDSA with less than one bit of nonce leakage. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 225–242, 2020.
2. N. Benger, J. Van de Pol, N. P. Smart, and Y. Yarom. ooh aah... just a little bit: A small amount of side channel can go a long way. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 75–92. Springer, 2014.

3. I. Biehl, B. Meyer, and V. Müller. Differential Fault Attacks on Elliptic Curve Cryptosystems. In *Advances in Cryptology-CRYPTO 2000*, pages 131–146. Springer, 2000.
4. B. B. Brumley and N. Tuveri. Remote timing attacks are still practical. In *European Symposium on Research in Computer Security*, pages 355–371. Springer, 2011.
5. W. Cao, J. Feng, H. Chen, S. Zhu, W. Wu, X. Han, and X. Zheng. Two lattice-based differential fault attacks against ECDSA with wNAF algorithm. In *ICISC 2015*, pages 297–313. Springer, 2015.
6. Certicom Research. Recommended Elliptic Curve Domain Parameters Standards for Efficient Cryptography (SEC) 2. https://www.iso.org/standard/76382.html, 2000.
7. Y. Chen and P. Q. Nguyen. BKZ 2.0: better lattice security estimates. In *Advances in Cryptology–ASIACRYPT 2011*, pages 1–20. Springer, 2011.
8. M. Ciet and M. Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *Designs Codes Cryptography*, 36(1):33–43, 2005.
9. D. Coppersmith, A. M. Odlzyko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1(1-4):1–15, 1986.
10. E. De Mulder, M. Hutter, M. E. Marson, and P. Pearson. Using bleichenbachers solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. *Journal of cryptographic engineering*, 4(1):33–45, 2014.
11. N. D. Elkies et al. Elliptic and modular curves over finite fields and related computational issues. *AMS IP STUDIES IN ADVANCED MATHEMATICS*, 7:21–76, 1998.
12. D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom. ECDSA key extraction from mobile devices via nonintrusive physical side channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1626–1638, 2016.
13. D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.
14. N. A. Howgrave-Graham and N. P. Smart. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography*, 23(3):283–290, 2001.
15. International Standard ISO/IEC 14888-3:2006(E). IT Security techniques Digital signatures with appendix Part 3: Discrete logarithm based mechanisms. https://www.iso.org/standard/76382.html, 2018.
16. D. Karaklajić, J.-M. Schmidt, and I. Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(12):2295–2306, 2013.
17. T. Kim and M. Tibouchi. Bit-flip faults on elliptic curve base fields, revisited. In *International Conference on Applied Cryptography and Network Security*, pages 163–180. Springer, 2014.
18. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
19. M. Ltd. Multiprecision Integer and Rational Arithmetic Cryptographic Library. https://github.com/miracl/MIRACL, 2019.
20. D. Micciancio and S. Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671. Springer, 2002.
21. D. Naccache, P. Q. Nguyên, M. Tunstall, and C. Whelan. Experimenting with Faults, Lattices and the DSA. In *International Workshop on Public Key Cryptography*, pages 16–28. Springer, 2005.
22. P. Q. Nguyen. Hermites constant and lattice algorithms. In *The LLL Algorithm: Survey and Applications*, pages 19–69. Springer, 2010.

23. P. Q. Nguyen and I. E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15(3), 2002.
24. P. Q. Nguyen and I. E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, codes and cryptography*, 30(2):201–217, 2003.
25. P. Q. Nguyen and M. Tibouchi. Lattice-based fault attacks on signatures. In *Fault Analysis in Cryptography*, pages 201–220. Springer, 2012.
26. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.). *IEEE Transactions on information Theory*, 24(1):106–110, 1978.
27. J. Schmidt and M. Medwed. A Fault Attack on ECDSA. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2009 Workshop on*, pages 93–99. IEEE, 2009.
28. C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994.
29. C.-P. Schnorr and H. H. Hörner. Attacking the chor-rivest cryptosystem by improved lattice reduction. In *Advances in Cryptology Eurocrypt 1995*, pages 1–12. Springer, 1995.
30. V. Shoup. Number Theory C++ Library (NTL) version 9.6.4. http://www.shoup.net/ntl/, 2016.
31. P. C. Van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of cryptology*, 12(1):1–28, 1999.