Lattice-based Fault Attacks against Deterministic Signatures ECDSA and EdDSA

Weiqiong Cao[1], Hongsong Shi[1], Hua Chen[2], Wei Xi[3], Haoyuan Li[2], Limin Fan[2] and Wenling Wu[2]

[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.

[3] Electric Power Research Institute, China Southern Power Grid, Guangzhou 510000, China.

Email: caoweqion@163.com, chenhua@tca.iscas.ac.cn

Abstract. Deterministic ECC-based signatures including deterministic ECDSA and EdDSA are becoming popular to be applied to blockchain and Internet of Things. Their security has received a considerable attention, and there have existed some differential fault attacks against them. However, the attacks have some problems such as high computational complexity and strict requirement of fault injection. In this paper eight efficient lattice-based fault attacks(and one differential fault attack) against deterministic ECDSA and two ones against EdDSA are proposed. All the fault models of such attacks are the random storage faults of intermediate values during signature, by which some faulty and one correct signatures are obtained to construct the models of lattice attacks(or the equations with two unknown) and thereby recover the private key. Unlike the previous differential fault attacks based on storage faults, our attacks do not need to guess the number and location of the faulty bits, and are still effective while the previous attacks are computationally infeasible. Moreover, compared with the previous lattice-based fault attacks against the non-deterministic signatures with random nonces, our attacks have more fault models besides the faulty nonce k, and only need random fault injection. We demonstrate the effectiveness of the attacks by simulations, which shows our attacks pose real threats to deterministic signature. The upper bound of the number of the faulty bits is just slightly less than the key length. We also discuss the corresponding countermeasures against our attacks.

Keywords: ECC, Fault Attack, Lattice Attack, Deterministic ECDSA, EdDSA

1 Introduction

Elliptic curve cryptosystem(ECC) [1] is one kind of the most popular public key cryptosystems. Compared with RSA, ECC is more suitable for embedded

device due to its shorter key length and faster execution, especially for Internet of Things(IoT). As a popular ECC algorithm, ECDSA signature scheme [2] is a variant of El-Gamal signature [3], in which there exists a random nonce k so as to generate random signature. However, not all the cryptographic devices have a good random number generator(RNG), such as IoT, RFID tags, and the softwares without RNG. If the nonces are reused in some signatures, the attacker can employ the signatures to recover the private key [4,5]. Hence, as the alternatives, the deterministic ECC-based signature schemes(DESS for short) are deployed in many cryptographic devices, especially in the devices without a good RNG, in which the nonces can be determined uniquely by the hash function of inputting message and private key.

Deterministic ECDSA(DECDSA) [6] on Weierstrass curve and EdDSA on Edwards curve are two types of the current common DESS. DECDSA as a variant of ECDSA has been specified in RFC 6979 [6] by Internet Engineering Task Force (IETF), and used to ensure the application security of blockchain. EdDSA is proposed by Bernstein et al. [7], and is also specified in RFC 8032 [8]. It has been proved that the speed of EdDSA on Curve25519 [9] is two times faster than the ECDSA recommended in NIST P-256 [10]. Therefore, EdDSA is becoming more popular in real application, and has been applied to many protocols including OpenSSH, Tor, TLS, Apple AirPlay, DNS protocols etc [11].

The standard DESS can resist the theoretical analysis, but their implementation faces the threat from physical attack, in which fault attack(FA) is one of the most frequently used physical attacks. FA exploits the faulty signatures caused by fault injection(FI) such as laser injection, strong electromagnetic radiation and glitches to deduce the secret key. At present, there are three typical fault attacks against ECC: weak curve fault attack(WCA) [12,13,14], differential fault attack(DFA) [12,15] and lattice-based fault attack(LFA) [4,5,15,16,17]. However, to our knowledge, only DFA is effective for DESS. Two kinds of DFAs against DESS are first proposed in IWSEC'2016 [11], including the faulty scalar multiplication(SM)-based DFA and the storage faults-based of intermediate values DFA. The first DFA assumes that some faults are induced during the calculation of SM, to get a faulty signature (r', s'). Meanwhile, the same message m can generate a correct signature (r, s). Thereby, the private key d_0 or d can be recovered by solving the equation $s - s' = d_0(r - r')$(EdDSA) or $s - s' = (s'r - r's)\, e^{-1}d$(DECDSA), where e is the hash value of message m. Unlike the first one, the second DFA assumes that a single bit or byte of the intermediate value(mv) is randomly tampered before the calculation of s, where mv can be the nonce k, the signature result r and so on. Consequently, mv is changed into $mv' = mv + \varepsilon$, where ε is a small random number. By constructing a differential distinguisher with mv and mv' the private key d_0 or d can be recovered. After that, the two types of DFAs are summarized in CT-RSA'2018 [18], and the second DFA is extended with more feasible targets of FI. Meanwhile, a practical DFA against the EdDSA signature between two virtual machines is introduced in Euros S&P'2018 [19], in which FI employs the Rowhammer attack. The result has also been specified in FIPS 186-5 (Draft).

In addition, more DFAs within the first type are introduced in FDTC'2017 [20] and AFRICACRYPT'2018 [21]. In conclusion, the first type of DFA can be implemented in practice. Nevertheless, it only has one available target of FI, and checking whether the input and output points during the calculation of SM are on the original curve is effective to resist this attack. For the second DFA, although there are more available targets of FI than the first DFA, the consequent requirement for FI becomes more strict due to needing a small amount of faulty bits of mv set for the attack. Otherwise, the attack is computationally infeasible.

To overcome the above problems of DFAs and expand the FAs against DESS, we research whether LFA can be applied to analyze DESS. LFA which combines FA and lattice attack(LA) is usually used for analyzing non-deterministic signature schemes with random nonces, and there have been many LFAs proposed in [4,5,15,16,17]. However, there seems no LFA against DESS in the current literature. To our knowledge, only one LFA against deterministic lattice signatures in lattice-based cryptography is proposed in [22], but it cannot be applied to analyze DESS because of the totally different structure of lattice-based cryptography with ECC.

Our contributions. In this paper, we found LFA still can be applied to DESS and even causes more enormous threats to DESS than to non-deterministic signatures. It is assumed that some random storage faults are induced into the intermediate values mv's in DESS. Thereby, we can build some models of lattice attack(LA), and first propose eight new LFAs(and one DFA) against DECDSA and two ones against EdDSA.

1. Compared with the previous DFAs based on the storage faults(the second type of DFA) against DESS, our attacks have more targets of FI, and recover the private key not by guess-determine analysis but by solving shortest vector problem(SVP) or closest vector problem(CVP). This reduces the requirement for FI and time complexity sharply so that our attacks can be implemented in practice. Even if the random faulty bits of mv's are slightly less than the key length, our attacks are still effective.

2. Different with the previous LFAs against non-deterministic signatures with random nonce k, our attacks are more feasible and aggressive. The faulty bits induced in our attacks are random, while the ones induced in the previous LFAs are controlled to be fixed or same. Moreover, our attacks are not only limited to construct CVP with the faulty nonce k as in the previous LFAs, but also construct more totally new SVPs with the other targets of FI.

3. The vulnerabilities of different algebraic structures of s in DECDSA and EdDSA for our attacks are analyzed, and it is concluded that the number of the coefficients multiplied by the secrets in the structure of s is proportional to the number of vulnerabilities.

4. The corresponding countermeasures in implementation are discussed, in which adding random number and double signatures are the optimal selections.

The remainder of the paper is organized as follows: Sect. 2 introduces the deterministic signature schemes DECDSA and EdDSA, and some theoretical basis about lattices, respectively. In Sect. 3, two representative kinds of LFAs against

DECDSA and EdDSA are presented in detail, and the other FAs are listed in Appendix A. Sect. 4 verifies the feasibility of the proposed attacks by simulations. The discussion about the corresponding countermeasures is deployed in Sect. 5. Finally, the conclusion is given in Sect. 6. Appendix B lists the verification algorithms of DECDSA and EdDSA.

2 Preliminaries

In this section, we first introduce the deterministic signature schemes, including the deterministic ECDSA signature on Weierstrass curve in prime field $F_p(p > 3)$ and the EdDSA signature on Edwards curve. Next, we give some theoretical basis about lattices.

2.1 Deterministic ECDSA Signature Scheme

The elliptic curve $E(a, b)$ is defined by Weierstrass equation in prime field F_p as follow

$$E(a, b) : y^2 = x^3 + ax + b \mod p, \tag{1}$$

where $a, b \in F_p$, and $4a^3 + 27b^2 \neq 0 \mod p$.

The additive group $E(F_p)$ consists of the set of points on $E(a, b)$ and the infinity point $\mathcal{O}$.

$$E(F_p) = \left\{(x, y)|x, y \in F_p, y^2 = x^3 + ax + b \bmod p\right\} \cup \{\mathcal{O}\}. \tag{2}$$

The scalar multiplication(SM) kG is the most elementary operation on $E(F_p)$ and is composed of some point doublings and additions. Given a basic point $G \in E(F_p)$ with order n and a scalar $k \in [0, n-1]$, the SM kG is defined as the sum of k G's. The security of ECC is based on the elliptic curve discrete logarithm problem(ECDLP): given the basic point $G \in E(F_p)$ and a point $Q \in E(F_p)$, it is hard to find the scalar $k \in [0, n-1]$ satisfying $Q = kG$.

As an alternative of ECDSA [2], deterministic ECDSA(**DECDSA**) has a similar procedure with ECDSA as shown in Algorithm 1. The only difference is that the nonce k is generated not by random number generator(RNG) but by an HMAC function $F(.)$ described in [23]. Since the hash value e of message m and private key d are the initial inputs of HMAC to generate k, the invariable message m in the signatures can generate same nonces k's.

2.2 Deterministic EdDSA Signature Scheme

EdDSA signature scheme is similar to DECDSA. It is proposed by Bernstein et al. [7] in view of high efficiency. The twisted Edwards curve $E(a, d)$ in finite field $F_q(q = 2^{255} - 19$ for curve25519) is defined as follow

$$E(a, d) : ax^2 + y^2 = 1 + dx^2y^2. \tag{3}$$

Where $a, d \in F_q$.

Algorithm 1 Deterministic ECDSA Signature

Require: message m, private key d.
Ensure: signature results r, s.
1: $e = SHA(m)$;
2: Generate $k \in [1, n-1] = F(e, d)$ with HMAC_PRNG(Difference: randomly generate k in ECDSA);
3: $Q(x_1, y_1) = kG$;
4: $r = x_1 \bmod n$;
5: **if** $r = 0$ **then** goto step 2;
6: $s = k^{-1}(e + dr) \bmod n$;
7: **if** $s = 0$ **then** goto step 2;
8: **return** (r, s)

Before signature, the b-bit private key d first derives two b-bit subkeys d_0 and d_1 by the hash function $H(d)$ with $2b$-bit output, where $H(d) = (h_0, h_1, ..., h_{2b-1})$, $d_0 = 2^{b-2} + \sum_{i=3}^{b-3} 2^i h_i$, and $d_1 = (h_b, ..., h_{2b-1})$. The public key P satisfies $P = d_0 G$, where G is the basis point with order n. The detailed signature procedure is described in Algorithm 2. Points R and P are encoded as the input of $H(.)$. Finally, the signer calculates to get the signature results $m, (R, s)$ and outputs them for verification.

Algorithm 2 Deterministic EdDSA Signature

Require: message m, subkeys (d_0, d_1).
Ensure: signature results R, s.
1: $k = H(d_1, m) \mod n$;
2: $R(x_1, y_1) = kG \in E_{a,d}(F_q)$;
3: $r = H(R, P, m) \mod n$;
4: $s = k + rd_0 \bmod n$;
5: **return** (R, s)

2.3 Lattices

In this section, we will give some theoretical basis about lattices. If the row vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_N \in \mathbb{R}^m$ of a matrix $M \in \mathbb{R}^{m \times N}$ are linearly independent, then the vector space $\mathcal{L}$ spanned by the vectors $\boldsymbol{b}_i$'s is the so-called lattice, where $\mathbb{R}^m$ is the m dimensional space in real number field $\mathbb{R}$. The lattice $\mathcal{L}$ satisfies

$$\mathcal{L} = \mathcal{L}(\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_N) = \{\boldsymbol{z} = \sum_{i=1}^{N} x_i \cdot \boldsymbol{b}_i | x_i \in \mathbb{Z}\}, \tag{4}$$

where $\boldsymbol{b}_i$'s is a basis of $\mathcal{L}$ and M is the basis matrix. For any $\boldsymbol{z} \in \mathcal{L}$, there exists $\boldsymbol{x} \in \mathbb{Z}^N$ so that $\boldsymbol{z} = \boldsymbol{x}M$. N is the dimension of $\mathcal{L}$. If m equals to N, then $\mathcal{L}$ is full rank. If $\boldsymbol{b}_i$ belongs to $\mathbb{Z}^m$ for any $i = 1, ..., N$, $\mathcal{L}$ is an integer lattice.

Shortest vector problem(SVP) and closest vector problem(CVP) are two famous hard problems in lattice. In order to weaken the hardness of SVP and CVP, approximate SVP(aSVP) and approximate CVP(aCVP) are introduced.

aSVP: given a basis $\boldsymbol{b}_i$'s of $\mathcal{L}$, find a nonzero vector $\boldsymbol{v} \in \mathcal{L}$ satisfying

$$\|\boldsymbol{v}\| \leq f(N)\lambda_1(\mathcal{L}). \tag{5}$$

Where $\lambda_1(\mathcal{L})$ is the length of the shortest vector and $f(N)$ is the approximate factor related to N. The norm $||\boldsymbol{v}||$ is defined as $\sqrt{{v_1}^2 + ... + {v_m}^2}$, i.e., $||\boldsymbol{v}||$ is the length of $\boldsymbol{v}$, where $\boldsymbol{v} = (v_1, ..., v_m)$, $v_i \in \mathbb{R}$. If $f(N)$ is the exponential form of N, then the aSVP with factor $f(N)$ can be solved using LLL algorithm [24] in polynomial time or LLL-based BKZ algorithms [25].

In addition, it has been proved [26] that a random lattice $\mathcal{L} \in \mathbb{R}^m$ with dimension N satisfies with overwhelming probability

$$\lambda_1(L) \approx \sqrt{\frac{N}{2\pi e}} vol(\mathcal{L})^{\frac{1}{N}}, \tag{6}$$

where $vol(\mathcal{L})$ is the determinant of $\mathcal{L}$.

aCVP: given a basis $\boldsymbol{b}_i$'s of $\mathcal{L}$ and a target vector $\boldsymbol{u} \in \mathbb{R}^m$, find a nonzero vector $\boldsymbol{v} \in \mathcal{L}$ satisfying

$$\|\boldsymbol{v} - \boldsymbol{u}\| = f(N)\lambda(\mathcal{L}, \boldsymbol{u}). \tag{7}$$

Where $\lambda(\mathcal{L}, \boldsymbol{u})$ is the closest distance from vector $\boldsymbol{u}$ to lattice $\mathcal{L}$.

Similarly, if $f(N)$ is the exponential form of N, the aCVP can be solved by using LLL-based Babai's nearest plane algorithm [27] in polynomial time. Furthermore, Babai has proved [27] that given a target vector $\boldsymbol{u}$, the lattice vector $\boldsymbol{v}$ can be determined in polynomial time when satisfying the inequation

$$\|\boldsymbol{v} - \boldsymbol{u}\| \leq c_1 ||b_N^*||^2 \leq \sqrt{\frac{N}{2\pi e}} vol(\mathcal{L})^{\frac{1}{N}}. \tag{8}$$

In addition, CVP is usually reduced into SVP by the embedding technique in practice [28] and SVP also can be transformed into a set of CVP oracle [29].

3 LFAs against Deterministic Signature Schemes

In this section, we will introduce the proposed lattice-based fault attacks.

In our attacks, all the fault models are the storage faults of intermediate values mv's in signatures, by which some faulty signatures can be obtained to construct the models of lattice attack(LA). Finally, the private key d or d_0 can be recovered by solving aCVP or aSVP in lattice. How to select the proper targets of FI for constructing the models of LA is the difficult point in our attacks. By

analysis, we found that the coefficients multiplied with the secrets d, d_0 and k are the key points constructing the models of LA. Once there exist some faults introduced into these coefficients, a LA can be carried on to reveal the private key. Hence, the final purpose of inducing faults into the targets of FI is to make the coefficients faulty, and the number of the coefficients decides the number of the targets of FI.

3.1 Targets of Fault Injection

Because EdDSA and DECDSA have different algebraic structures, they have different secrets, the corresponding coefficients and targets of FI. We compare the algebraic structures in DECDSA and EdDSA as follow, respectively.

$$\begin{aligned} s &= k + rd_0 \mod n\,(EdDSA) \\ s &= k^{-1}\,(e + rd) \mod n\,(DECDSA). \end{aligned} \tag{9}$$

From the equations, only the secret d_0 in EdDSA has one coefficient r, while the two secrets d and k^{-1} in DECDSA have more coefficients besides r. The secrets, the corresponding coefficients and the targets of FI, and the solved problems for our attacks in DECDSA and EdDSA are summarized in Table 1. For the secret d in DECDSA, there exist two coefficients r and k^{-1}. If the two targets r, k corresponding to the coefficient r are partially tampered by FI, r will be changed into r' by which the model of lattice attack can be constructed. Thereby, the secret can be recovered by solving aSVP or aCVP. Nevertheless, it is worth noting that the secret is recovered not by solving aSVP but by solving the binary linear equations when the target x_1 are tampered as shown in Table 1. This is because the faulty r' is known and the nonce k is fixed, and thereby the secret can be directly recovered by solving the equations as the previous DFAs. Analogously, the coefficient k^{-1} multiplied by d in DECDSA corresponds to two targets, and the coefficient $e + rd$ multiplied by the secret k^{-1} in DECDSA corresponds to four targets, while the coefficient r multiplied by the secret d_0 in EdDSA corresponds to two targets. In a word, there are nine targets of FI in DECDSA and two ones in EdDSA. Meanwhile, except the attacks targeting k before the calculation of SM kG(aCVP) and targeting x_i before the calculation of r(binary equations), the other attacks recover the private key by solving aSVP.

Table 1. The secrets, coefficients, targets and solved problem in our attacks against DECDSA/EdDSA.

Algorithms \ Item	secrets	coefficients	targets of FI	solved problem
DECDSA	d	r	r during the calculation of s	aSVP
			k before the calculation of SM kG	aCVP
			x_i before the calculation of r	binary equations
		k^{-1}	k^{-1} during the calculation of s	aSVP
			k during the calculation of s	aSVP
	k^{-1}	$e+rd$	d during the calculation of s	aSVP
			e during the calculation of s	aSVP
			rd during the calculation of s	aSVP
			$e+rd$ during the calculation of s	aSVP
EdDSA	d_0	r	r during the calculation of s	aSVP
			k before the calculation of SM kG	aCVP

As a result, EdDSA has less targets of FI than DECDSA. Although the algebraic structure of s in EdDSA is simpler and has no modulo inverse operation, it shows more better performance to resist our LFAs than the general algebraic structures on the contrary.

Based on the above target of FI, the following sections introduce two typical kinds of LFAs in our attacks, and the others are detailed in the Appendix A.

All our LFAs consist of two steps. First, fault attack is carried out to change the intermediate value mv into $mv' = mv + \varepsilon$, where ε is a random number so that the low $w(w > 0)$ bits of mv are disturbed. Hence, ε satisfies $-2^w < \varepsilon < 2^w$ $(w > 0)$. That is, $\varepsilon \bmod n \in (-2^w + n, n) \cup [0, 2^w)$. Next, a model of lattice attack based on the random ε is built to reveal d in DECDSA or d_0 in EdDSA.

3.2 LFA on r before the calculation of s by solving aSVP

DECDSA.Step 1: $r \to r_i = r + \varepsilon_i$ by fault attack.

When r is loaded to calculate the signature result s(line 6 in Algorithm 2), a fault is induced into r in $N-1$ signatures. r is rewritten into $r_i = r + \varepsilon_i$ for $i = 1, ..., N-1$, where ε_i satisfying $-2^w < \varepsilon_i < 2^w$ is a random number so that the low $w(w > 0)$ bits of r are disturbed. The valid signature value (r, s_0) and $N-1$ invalid ones (r_i, s_i) for the same input message m can be represented as

$$\begin{cases} s_0 = k^{-1}(e + rd) \mod n \\ s_i = k^{-1}(e + (r + \varepsilon_i)d) \mod n (i = 1, ..., N-1). \end{cases} \tag{10}$$

Step 2: recover the private key d by lattice attack.

After reduction, the equations (10) can be transformed as

$$\varepsilon_i = (s_i - s_0)\, d^{-1} k \mod n \tag{11}$$

Assuming that $A_i = s_i - s_0 \bmod n$, $D = d^{-1}k \mod n$, there must exist $h_i \in \mathbb{Z}$ for $i = 1, ..., N-1$ so that

$$\varepsilon_i = A_i D + h_i n, \tag{12}$$

where D is a fixed value due to the same message m.

Since $-2^w < \varepsilon_i < 2^w$, we have

$$|A_i D + h_i n| < 2^w. \tag{13}$$

For $i = 1, \ldots, N-1$, let the basis matrix $M \in \mathbb{Z}^{N\times N}$ of lattice $\mathcal{L}$ satifying

$$M = \begin{pmatrix} n & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & n & 0 \\ A_1 & \cdots & A_{N-1} & 2^w/n \end{pmatrix},$$

and $\boldsymbol{x} = (h_1, \ldots, h_{N-1}, D) \in \mathbb{Z}^N$, then the nonzero lattice vector $\boldsymbol{v} = \boldsymbol{x}M = (A_1 D + h_1 n, \ldots, A_{N-1} D + h_{N-1} n, D2^w/n)$.

From the inequations (13), if $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f+\log\sqrt{2\pi e}}{f-w-\log\sqrt{2\pi e}}$, then the vector $\boldsymbol{v}$ satisfies

$$\|\boldsymbol{v}\| < \sqrt{N}2^w < \sqrt{\frac{N}{2\pi e}} vol(\mathcal{L})^{\frac{1}{N}}, \tag{14}$$

where $vol(\mathcal{L}) = det(M) = n^{N-2}2^w$ and $f = \lceil \log n \rceil$.

As mentioned in Sect. 2.3, $\mathcal{L}$ behaves like a random lattice, so the inequality (14) can be viewed as a SVP in $\mathcal{L}$. Hence, we can obtain the short vector $\boldsymbol{v}$ by LLL reduction, and thereby get the value of D. Consequently, the private key $d = (Ds_0 - r)^{-1}e \bmod n$ can be deduced by substituting D into equations (10).

EdDSA. In all the DFAs against EdDSA proposed in [18,19,20,21,30], the recovered key is not the private key d but the sub-keys d_0 or d_1. Similarly, only the sub-key d_0 is recovered in our attacks, but the forged signatures with the derived d_0 still can successfully pass verification, in which k is not the value derived by message and d_0 but an arbitrary value.

Similar with the attack against DECDSA, FA is first carried on to obtain

$$\begin{cases} s_0 = k + rd_0 \mod n \\ s_i = k + (r + \varepsilon_i)d_0 \mod n (i = 1, ..., N-1). \end{cases} \tag{15}$$

After reduction, the equations (15) can be transformed as

$$\varepsilon_i = A_i D + h_i n, \tag{16}$$

where $A_i = s_i - s_0 \mod n$, $D = d_0^{-1} \mod n$, $h_i \in \mathbb{Z}$.

Since $-2^w < \varepsilon_i < 2^w$, we have

$$|A_i D + h_i n| < 2^w. \tag{17}$$

Analogously, from the inequations (17), a SVP in a $\mathcal{L}$ can be constructed. If $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f+\log\sqrt{2\pi e}}{f-w-\log\sqrt{2\pi e}}$, d_0 can be recovered by solving aSVP.

3.3 LFA on k before the calculation of SM kG by solving aCVP

DECDSA. Step 1: disturb k into $k_i = k + \varepsilon_i$ by fault attack.

During DECDSA signature, the nonce k is generated by $F(d, m)$(line 2 in Algorithm 2), and is loaded to calculate SM kG with the base point G. If some faults are induced during the loading in $N-1$ signatures, then k is rewritten into $k_i = k + \varepsilon_i$ for $i = 1, ..., N-1$, , where $-2^w < \varepsilon_i < 2^w$. Finally, we obtain a valid signature result (r_0, s_0) and $N-1$ invalid (r_i, s_i) for the same message m. The final equations are as follow

$$\begin{cases} k = {s_0}^{-1}(e + r_0 d) \bmod n \\ k + \varepsilon_i = {s_i}^{-1}(e + r_i d) \bmod n (i = 1, ..., N-1). \end{cases} \tag{18}$$

Step 2: recover the private key d by lattice attack.

After reduction, the equations (18) can be transformed as

$$\varepsilon_i = \left({s_i}^{-1} r_i - {s_0}^{-1} r_0\right) d - \left({s_0}^{-1} - {s_i}^{-1}\right) e \bmod n. \tag{19}$$

Let $A_i = \left({s_i}^{-1} r_i - {s_0}^{-1} r_0\right) \bmod n$, $B_i = ({s_0}^{-1} - {s_i}^{-1}) e \bmod n$, then there must exist $h_i \in \mathbb{Z}$ satisfying

$$\varepsilon_i = A_i d + h_i n - B_i. \tag{20}$$

Since $-2^w < \varepsilon_i < 2^w$, we have

$$|A_i d + h_i n - B_i| < 2^w. \tag{21}$$

A model of LA can be built by the inequations (21). For $i = 1, \ldots, N-1$, a lattice $\mathcal{L}$ can be spanned by the row vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N$ of matrix

$$M = \begin{pmatrix} n & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & n & 0 \\ A_1 & \cdots & A_{N-1} & 2^w/n \end{pmatrix}.$$

Let $\boldsymbol{x} = (h_1, \ldots, h_{N-1}, d) \in \mathbb{Z}^N$, then the nonzero lattice vector $\boldsymbol{v} = \boldsymbol{x}M$ equals to $(A_1 d + h_1 n, \ldots, A_{N-1} d + h_{N-1} n, d 2^w/n)$. In addition, we define the non-lattice vector $\boldsymbol{u} = (B_1, \ldots, B_{N-1}, 0) \in \mathbb{Z}^N$. Naturally, from the inequations (21), we have

$$\|\boldsymbol{v} - \boldsymbol{u}\| < \sqrt{N} 2^w. \tag{22}$$

If $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f + \log\sqrt{2\pi e}}{f - w - \log\sqrt{2\pi e}}$, then

$$\|\boldsymbol{v} - \boldsymbol{u}\| < \sqrt{N} 2^w < \sqrt{\frac{N}{2\pi e}} vol(\mathcal{L})^{\frac{1}{N}}, \tag{23}$$

where $vol(\mathcal{L}) = det(M) = n^{N-2} 2^w$, and $f = \lceil \log n \rceil$.

According to the equation (8) mentioned in Sect. 2.3, the inequality (23) can be view as a CVP in $\mathcal{L}$. We can obtain $\boldsymbol{v}$ by LLL-based BaBai algorithm, and deduce the private key d from $\boldsymbol{v}$.

EdDSA. It is assumed that k derived from the same m is changed into $k_i = k + \varepsilon_i(\ i = 1, ..., N-1)$by FA before the calculation of SM kG. The valid signature results (R_0, s_0) and the invalid ones (R_i, s_i) are represented as

$$\begin{cases} s_0 = k + r_0 d_0 \bmod n \\ s_i = k + \varepsilon_i + r_i d_0 \bmod n (i = 1, ..., N-1), \end{cases} \tag{24}$$

where $r_i = H(R_i, P, m)$ and $-2^w < \varepsilon_i < 2^w$ for $i = 0, ..., N-1$.

After several iterations, the equations (24) can be transformed into

$$\varepsilon_i = A_i d_0 - B_i + h_i n, \tag{25}$$

where $h_i \in \mathbb{Z}$, $A_i = (r_0 - r_i) \bmod n$, $B_i = (s_0 - s_i) \bmod n$ for $i = 1, ..., N-1$.

Since $-2^w < \varepsilon_i < 2^w$, we have

$$|A_i d_0 - B_i + h_i n| < 2^w. \tag{26}$$

By the same way, a model of LA can be built. If $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f + \log\sqrt{2\pi e}}{f - w - \log\sqrt{2\pi e}}$, the subkey d_0 can be recovered by solving aCVP in lattice.

In conclusion, regardless of solving aSVP or aCVP, the number w of faulty bits and the number N of signatures needed in all the attacks satisfy $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f + \log\sqrt{2\pi e}}{f - w - \log\sqrt{2\pi e}}$. The smaller w, the smaller needed number N of signature. Moreover, the upper bound of w can be even up to $f - \lceil \log\sqrt{2\pi e} \rceil - 1$.

4 Attack Simulations and Complexity Comparisons

In this section we do some simulations to demonstrate the proposed LFAs. The attack based on the faulty target x_i is not simulated due to the simple way to solve the equations.

First, we target 256-bit DECDSA and EdDSA to simulate the fault attacks. We randomly select a message m and a private key d to generate a correct DECDSA/EdDSA signature results. Next, based the proposed fault models, forge $N-1$ faulty signature results for the fixed m and d. In every forged signature, the lower w bits of intermediate value(mv) is randomized by a w-bit random number $\zeta_i (i = 1, ..., N-1)$. That is, $v_i = v \oplus \zeta_i = v + \varepsilon_i$, where ε_i is derived by ζ_i and its absolute value is less than 2^w. w and N can be set to different values. Finally, we carry on the proposed lattice attacks with the forged and correct signatures to recover the private key d(or d_0) in DECDSA(or EdDSA). If the derived d(or d_0) satisfies that dG(or $d_0 G$) equals to the public key P or a new signature with inputting d(or d_0) can pass the verification in DECDSA/EdDSA, the attack is considered to be successful.

Table 2. The number of signatures and success probability for the lattice attacks against DECDSA based on different w.

data / fault target	$w_{max}=251$		$w=245$		$w=190$		$w=128$		$w=124$	
	N	γ	N	γ	N	γ	N	γ	N	γ
r during the calculation of s	90	87%	29	96%	6	79%	3	42%	3	66%
k before the calculation of SM kG	200	6%	29	96%	6	100%	3	55%	3	100%
k^{-1} during the calculation of s	90	60%	29	97%	6	52%	3	45%	3	62%
k during the calculation of s	90	33%	29	97%	6	82%	3	62%	3	44%
$e, rd, e+rd$ during the calculation of s	90	25%	29	97%	6	66%	3	43%	3	52%
d during the calculation of s	90	100%	29	98%	6	60%	3	42%	3	49%

Table 3. The number of signatures and success probability for the lattice attacks against EdDSA based on different w.

data / fault target	$w_{max}=251$		$w=245$		$w=190$		$w=128$		$w=124$	
	N	γ	N	γ	N	γ	N	γ	N	γ
r before calculating s	90	100%	29	100%	5	87%	3	46%	3	57%
k before ECSM kG	100	70%	29	96%	5	100%	3	20%	3	100%

In experiments, we employ LLL algorithm and LLL-based BaBai algorithm implemented in NTL library [31] to solve aSVP and aCVP, respectively. As stated above, when $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f+\log\sqrt{2\pi e}}{f-w-\log\sqrt{2\pi e}}$, the correct d(or d_0) can be recovered by the lattice attacks. If $f = 256$, then w can be up to 253 and $N > 271$ in theory. However, the upper bound of w is slightly smaller than the theoretical value in our experiments, because the dimension $N > 271$ needed in the constructed lattice($w = 253$) is too big to make the attack feasible computationally. In addition, as introduced in Sect. 2.3, the reduced vectors by LLL algorithm are just approximate to the shortest(or closest) vector with the exponential factor $2^{(N-1)/2}$, when the parameter δ equals to 3/4. If N exceeds a certain bound, there is a big gap between the LLL-reduced short vectors and the shortest vector. Hence, there exists a success rate γ for LLL reduction.

Table 2 and Table 3 list the number N of signature and the corresponding success rate γ in the lattice attacks against DECDSA and EdDSA when w is different value, respectively. Because the faulty targets $e, rd, e+rd$ of FI can lead to the same model of LA, we just implement the simulation of LFA based on faulty e. The experiments show that γ is greater than 0 only when $w \leq 251$, i.e, the upper bound of w is 251 which is slightly smaller than the key bit length($f = 256$). When $w = 251$, the success rate γ of solving aCVP is less than that of solving aSVP. Moreover, when $w = 128$, only three signature results($N = 3$, two faulty results and one correct results) are needed to recover the private key d/d_0 successfully, which is just 1 greater than the ones in the previous DFAs. Nevertheless, it is impossible for the previous attacks based on storage faults to recover the private key when $w = 128$.

Table 4. Our attacks VS the previous DFAs based on storage faults.

Item \ Fault attack	Our attacks	The previous attacks
Method	solving aSVP/aCVP	guess-determine
Upper bound of w	251	≈ 64
Time complexity($\mathbb{C}$)	$O\left(\left(2+\frac{f+\log\sqrt{2\pi e}}{f-w-\log\sqrt{2\pi e}}\right)^5(\log A)^2\right)$	$O(2^w)$
Attack time($w=251$)	≈ 64 s	$\approx O(2^{251})$(unfeasible)
Attack time($w=128$)	$1\sim 2$ ms	$O(2^{128})$(unfeasible)
$N(w=128)$	3	2

Some comparisons between our LFAs and the previous DFAs based on storage faults are listed in Table 4. LLL reduction takes up the main time consumption in our LFAs, and its time complexity $\mathbb{C}$ in the worst case is equal to $O\left(N^5(\log A)^2\right)$, i.e., $O\left(\left(2+\frac{f+\log\sqrt{2\pi e}}{f-w-\log\sqrt{2\pi e}}\right)^5(\log A)^2\right)$, where A is the maximum length in the original lattice vectors. $\mathbb{C}$ is proportional to w and is polynomial time. By contrast, the time complexity of previous DFAs in the worst case is equal to $O\left(2^w\right)$, which is exponential time and infeasible computationally when $w > 64$.

Obviously, our attacks have more advantages at time complexity, and reduce the requirement of FI sharply due to the sufficiently big w-bit random faults.

5 Countermeasures

In this section, we discuss the effectiveness of the general countermeasures against our attacks.

-**Using random number**. It has been implemented in EdDSA and XEdDSA signature schemes [32,33]. When adding a random number α as the input of the function of calculating k, each signature is different for the same message m and can pass the verification without any modification. Our attacks require many repeated signatures with the fixed m, while the randomness of signature breaks this condition. Hence, adding random number in DESS can prevent all our attacks. However, a random number generator is also brought back in signature, which deviates from the original intention of designing deterministic signature. Moreover, if the random number is reused, our attacks still work. If there exist some bits of k to be leaked, this returns to the same problem faced by non-deterministic signature with random nonce again.

-**Check code**. If an odd-even check or CRC check is deployed when writing/reading the storages, our attacks are still effective. We can implement the fault injection just after the check, which is feasible in practice.

-**Verification before outputting the signature results**. If the verification before output is passed, then output the signature results, otherwise return error. In our LFAs, only the attack whose target of FI is the nonce k before the

calculation of SM kG can resist this countermeasure. Nevertheless, the calculation of verification takes more time than signature, which is inefficient.

-**Double computation and comparison**. Calculate two times signatures with same message and compare whether the two signature results are consistent before output. If the results are inconsistent, return error. This is effective for resisting all our attacks. It seems the only practical countermeasure when there is no good random number generator in the cryptographic devices and only one target of FI can be induced. However, it also has the same inefficient problem with the above verification countermeasure.

-**Infective computation**. [20] introduces a fault infective computation, in which the hash function of generating r in EdDSA is processed two times. Consequently, there are two results r_1 and r_2 which equal to r. If there exist some faults during the calculation of r, the result s is also diffused by a random number t. That is, $s = k + r_1 + (d-t)r_1 + (t-1)r_2 \bmod n$, where t is changed in each signature. If $r_1 \neq r_2$, the output s is not the correct one and can not be used for attack. Similarly, the attack targeting $k(k \to k')$ before the calculation of SM still effective against this countermeasure. Because the two computations of hash functions obtain two same faulty r', then $s = k' + r' + (d-t)r' + (t-1)r' = k' + r'd \bmod n$. Hence, s is still available for the attack.

To sum up, if there exists a good random number generator in the implementation of DESS, adding random number seems the most effective method to protect the deterministic signature from all our LFAs. However, it should be ensured that the introduced random number has good randomness and is used properly, otherwise, our attacks still pose threats to DESS. In addition, without regard to the efficiency of implementation, double computation and comparison for signature is another priority selection.

6 Conclusion

In this paper, we present eight new LFAs (and one DFA) against deterministic ECDSA and two ones against EdDSA, respectively. All the fault models are the random storage faults of the intermediate values mv's during signature. Some random faults are first introduced into mv by fault attack, and then the private key d or d_0 can be recovered by lattice attack or solving equations. Consequently, we find that DECDSA has more vulnerabilities over EdDSA due to the algebraic structure of s in signature. The simulations and the complexity comparisons confirm the effectiveness of our attacks. For a 256-bit standard DECDSA(or EdDSA), the upper bound w of the random faulty bits in our LFAs is up to 251. Moreover, two faulty signatures are sufficient to recover the private key d(or d_0) when $w = 128$. Finally, we discuss the general countermeasures against our attacks. Double computation and comparison for signature and adding random number are the priority selections. This is the first time that we introduce the LFAs with random storage faults to enrich the fault attacks against DESS. The LFAs reduce the time complexity and the requirements of FI by adding signature data.

Further Work. For the other types of signature schemes, whether there are more vulnerable points due to their algebraic structures is needed to be further studied. In addition, If the nonce is misused or there is no good RNG in the hedged signature between non-deterministic and deterministic signatures, whether our attacks are still effective requires further analysis. Finally, in view of the excellent feasibility of our attacks, our future work will focus on the practical LFAs against the implementation of DESS.

References

1. Miller, V.: Use of Elliptic Curves in Cryptography. In: Advances in Cryptology-CRYPTO'85 Proceedings, Springer (1986) 417–426
2. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International Journal of Information Security **1**(1) (2001) 36–63
3. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans.inf.theory **31**(4) (1984) 469–472
4. Naccache, D., Nguyên, P.Q., Tunstall, M., Whelan, C.: Experimenting with Faults, Lattices and the DSA. Springer Berlin Heidelberg (2005)
5. Cao, W., Feng, J., Zhu, S., Chen, H., Wu, W., Han, X., Zheng, X.: Practical lattice-based fault attack and countermeasure on sm2 signature algorithm. In: International Conference on Information and Communications Security. (2015) 62–70
6. Pornin, T.: ministic usage of the digital signature algorithm (dsa) and elliptic curve digital signature algorithm. RFC 6979 (2013)
7. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of Cryptographic Engineering **2**(2) (2012) 77–89
8. Josefsson, S., Liusvaara, I.: Edwards-curve digital signature algorithm. RFC 8032 (2017)
9. Bernstein, D.J.: Curve25519: New diffie-hellman speed records. Lecture Notes in Computer Science **3958**(1) (2006) 207–228
10. Gueron, S., Krasnov, V.: Fast prime field elliptic-curve cryptography with 256-bit primes. Journal of Cryptographic Engineering **5**(2) (2015) 141–151
11. : Things that use ed25519. https://ianix.com/pub/ed25519-deployment.html (2017)
12. Biehl, I., Meyer, B., Müller, V.: Differential Fault Attacks on Elliptic Curve Cryptosystems. In: Advances in Cryptology-CRYPTO 2000, Springer (2000) 131–146
13. Jager, T., Schwenk, J., Somorovsky, J.: Practical invalid curve attacks on tls-ecdh. In: European Symposium on research in computer security, Springer (2015) 407–425
14. Takahashi, A., Tibouchi, M.: Degenerate fault attacks on elliptic curve parameters in openssl. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE (2019) 371–386
15. Schmidt, J., Medwed, M.: A fault attack on ecdsa. In: The Workshop on Fault Diagnosis and Tolerance in Cryptography. (2009) 93–99
16. Nguyen, P.Q., Tibouchi, M.: Lattice-Based Fault Attacks on Signatures. (2013)
17. Cao, W., Feng, J., Chen, H., Zhu, S., Wu, W., Han, X., Zheng, X.: Two lattice-based differential fault attacks against ecdsa with wnaf algorithm. In: International Conference on Information Security and Cryptology. (2015) 297–313

18. Ambrose, C., Bos, J.W., Fay, B., Joye, M., Lochter, M., Murray, B.: Differential attacks on deterministic signatures. In: Cryptographers Track at the RSA Conference. (2018) 339–353
19. Poddebniak, D., Schinzel, S., Somorovsky, J., Lochter, M., Roesler, P.: Attacking deterministic signature schemes using fault attacks. In: EuroS&P 2018. IEEE Computer Society. (2018)
20. Romailler, Y., Pelissier, S.: Practical fault attack against the ed25519 and eddsa signature schemes. In: The Workshop on Fault Diagnosis & Tolerance in Cryptography. (2017) 17–24
21. Samwel, N., Batina, L.: Practical fault injection on deterministic signatures: The case of eddsa. In: International Conference on Cryptology in Africa. (2018) 306–321
22. Bruinderink, L.G., Pessl, P.: Differential fault attacks on deterministic lattice signatures. In: IACR Transactions on Cryptographic Hardware and Embedded Systems (2018). (2018) 21–43
23. Barker, E.B., Kelsey, J.M.: SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. National Institute of Standards & Technology (2012)
24. Lenstra, H.W., Lenstra, A.K., Lovfiasz, L.: Factoring polynomials with rational coeficients. In: Mathematische Ann. (1982) 515–534
25. Schnorr, C.P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theoretical Computer Science **53**(2-3) (1987) 201–224
26. M, A.: Generating random lattices according to the invariant distribution. Draft of March (2006)
27. Babai, L.: On lovász' lattice reduction and the nearest lattice point problem (shortened version). Combinatorica **6**(1) (1986) 1–13
28. Nguyen, P.Q., Stern, J.: Lattice reduction in cryptology: An update. Lecture Notes in Computer Science **1838** (2000) 85–112
29. Goldreich, O., Micciancio, D., Safra, S., Seifert, J.P.: Approximating shortest lattice vectors is not harder than approximating closet lattice vectors. Information Processing Letters **71**(2) (1999) 55–61
30. Barenghi, A., Gerardo, P.: A note on fault attacks against deterministic signature schemes. In: International Workshop on Security. (2016) 182–192
31. Shoup, V.: Number Theory C++ Library (NTL) version 9.6.4. http://www.shoup.net/ntl/ (2016)
32. Susella, R.: Breaking ed25519 in wolfssl. In: Topics in CryptologyCCT-RSA 2018: The Cryptographers' Track at the RSA Conference 2018. (2018)
33. Perrin, T.: The xeddsa and vxeddsa signature schemes. Unpublished manuscript (2016)

A Appendix A

This appendix lists the remaining proposed attacks against DECDSA, including the LFAs on k, k^{-1}, e, rd, $e+rd$ and d during the calculation of s and the DFA on x_i before the calculation of r. EdDSA has not these vulnerabilities due to its algebraic structure.

A.1 LFA on k during the calculation of s by solving aSVP

DECDSA. Step 1: $k \to k_i = k + \varepsilon_i$ by fault attack.

When k is loaded to calculate the signature result s in Line 6 of Algorithm 2, a fault is induced into k in $N-1$ signatures. k is rewritten into $k_i = k + \varepsilon_i$ for $i = 1, ..., N-1$, where $-2^w < \varepsilon_i < 2^w (w > 0)$ is a random number so that the lower w bits of k are disturbed. The valid signature value (r, s_0) and $N-1$ invalid ones (r, s_i) for the same input message m are represented as

$$\begin{cases} k = {s_0}^{-1} (e + rd) \bmod n \\ k + \varepsilon_i = {s_i}^{-1} (e + rd) \bmod n (i = 1, ..., N-1) \end{cases}. \tag{27}$$

Step 2: recover the private key d by lattice attack.

After reduction, the equations (27) can be transformed as

$$\varepsilon_i = \left({s_i}^{-1} - {s_0}^{-1}\right) (e + rd) \bmod n. \tag{28}$$

Assuming that $A_i = {s_i}^{-1} - {s_0}^{-1} \bmod n$, $D = e + rd \mod n$, there must exist $h_i \in \mathbb{Z}$ for $i = 1, ..., N-1$ satisfying

$$|A_i D + h_i n| < 2^w, \tag{29}$$

where D is a fixed value due to the same message m.

For $i = 1, \ldots, N-1$, let the basis matrix of lattice $\mathcal{L}$

$$M = \begin{pmatrix} n & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & n & 0 \\ A_1 & \cdots & A_{N-1} & 2^w/n \end{pmatrix},$$

$\boldsymbol{x} = (h_1, \ldots, h_{N-1}, D) \in \mathbb{Z}^N$, then the nonzero lattice vector $\boldsymbol{v} = \boldsymbol{x}M = (A_1 D + h_1 n, \ldots, A_{N-1} D + h_{N-1} n, D 2^w/n)$.

If $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f + \log\sqrt{2\pi e}}{f - w - \log\sqrt{2\pi e}}$, we have

$$\|\boldsymbol{v}\| < \sqrt{N} 2^w < \sqrt{\frac{N}{2\pi e}} vol(\mathcal{L})^{\frac{1}{N}}, \tag{30}$$

where $vol(\mathcal{L}) = det(M) = n^{N-2} 2^w$.

As mentioned in Sect. 2, $\mathcal{L}$ behaves like a random lattice, so the inequation (30) can be viewed as a SVP in $\mathcal{L}$. Hence, we can recover D by solving aSVP, and deduce the private key d.

A.2 LFA on k^{-1} during the calculation of s by solving aSVP

DECDSA. Step 1: $K = k^{-1} \bmod n \to K_i = k^{-1} \bmod n + \varepsilon_i$ by fault attack.

When K derived from $k^{-1} \bmod n$ is loaded to calculate the signature result s in Line 6 of Algorithm 2, a fault is induced into K in $N-1$ signatures. K is rewritten into $K_i = K + \varepsilon_i$ for $i = 1, ..., N-1$, where $-2^w < \varepsilon_i < 2^w (w > 0)$ is a random number so that the lower w bits of k are disturbed. The valid signature

value (r, s_0) and $N-1$ invalid ones (r, s_i) for the same input message m are represented as

$$\begin{cases} s_0 = K\,(e + rd) \bmod n \\ s_i = (K + \varepsilon_i)\,(e + rd) \bmod n (i = 1, ..., N-1). \end{cases} \tag{31}$$

Step 2: recover the private key d by lattice attack.

After reduction, the equations (31) can be transformed as

$$\varepsilon_i = A_i D + h_i n, \tag{32}$$

where $A_i = s_i - s_0 \mod n$, $D = (e + rd)^{-1} \bmod n$(fixed value), $h_i \in \mathbb{Z}$.

Since $-2^w < \varepsilon_i < 2^w$, we have

$$|A_i D + h_i n| < 2^w. \tag{33}$$

Similarly, from the inequations (33), a SVP in a $\mathcal{L}$ can be constructed. If $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f + \log\sqrt{2\pi e}}{f - w - \log\sqrt{2\pi e}}$, we can recover D by solving aSVP and deduce the private key d.

A.3 LFA on d during the calculation of s by solving aSVP

DECDSA. Step 1: $d \to d_i = d + \varepsilon_i$ by fault attack.

When d is loaded to calculate the signature result s, a fault is induced into d in $N-1$ signatures. d is rewritten into $d_i = d + \varepsilon_i$ for $i = 1, ..., N-1$, where $-2^w < \varepsilon_i < 2^w (w > 0)$ is a random number so that the lower w bits of k are disturbed. The valid signature value (r, s_0) and $N-1$ invalid ones (r, s_i) for the same input message m are represented

$$\begin{cases} s_0 = k^{-1}\,(e + rd) \bmod n \\ s_i = k^{-1}\,(e + r(d + \varepsilon_i)) \bmod n (i = 1, ..., N-1). \end{cases} \tag{34}$$

Step 2: recover the private key d by lattice attack.

After reduction, the equations (34) can be transformed as

$$\varepsilon_i = A_i k + h_i n, \tag{35}$$

where $A_i = r^{-1}(s_i - s_0) \mod n$, $h_i \in \mathbb{Z}$.

Since $-2^w < \varepsilon_i < 2^w$, we have

$$|A_i D + h_i n| < 2^w. \tag{36}$$

Similarly, from the inequations (36), if $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f + \log\sqrt{2\pi e}}{f - w - \log\sqrt{2\pi e}}$, we can recover k by solving aSVP and deduce the private key d.

A.4 LFA on e, rd and $e + rd$ during the calculation of s by solving aSVP

All the targets e, rd and $e + rd$ correspond to the coefficient $e + rd$. If the lower w bits of the three targets are disturbed by FI, then a same model of LA can be built. Therefore, we just define mv as any one of the three targets(intermediate value), that is, mv can be e, rd or $e + rd$.

DECDSA. Step 1: $mv \rightarrow mv_i = mv + \varepsilon_i$ by fault attack.

When mv is loaded to calculate the signature result s in Line 6 of Algorithm 2, a fault is induced into mv in $N-1$ signatures. mv is rewritten into $mv_i = mv+\varepsilon_i$ for $i = 1, ..., N-1$, where $-2^w < \varepsilon_i < 2^w (w > 0)$ is a random number so that the lower w bits of mv are disturbed. The valid signature value (r, s_0) and $N-1$ invalid ones (r, s_i) for the same input message m represent

$$\begin{cases} s_0 = k^{-1}(e + rd) \mod n \\ s_i = k^{-1}(e + rd + \varepsilon_i) \mod n (i = 1, ..., N-1). \end{cases} \tag{37}$$

Step 2: recover the private key d by lattice attack.

After reduction, the equations (37) can be transformed as

$$\varepsilon_i = A_i k + h_i n, \tag{38}$$

where $A_i = s_i - s_0 \mod n$, $h_i \in \mathbb{Z}$.

Since $-2^w < \varepsilon_i < 2^w$, we have

$$|A_i D + h_i n| < 2^w. \tag{39}$$

Similarly, from the inequations (39), if $w < f - \log\sqrt{2\pi e}$ and $N > 1 + \frac{f+\log\sqrt{2\pi e}}{f-w-\log\sqrt{2\pi e}}$, we can recover k by solving SVP and deduce the private key d.

A.5 DFA on x_i before the calculation of r by solving equations

After the calculation of SM kG, x_i will be reduced into r by $r = x_i \mod n$. If some random faults are induced into x_i in signature, then x_i and r are changed into x_i' and r', respectively. There is no need to control the number of faulty bits. Consequently, we can obtain the correct signature (r, s) and the faulty signature (r', s') for a fixed message m. The signatures are represented as

$$\begin{cases} s = k^{-1}(e + rd) \mod n \\ s' = k^{-1}(e + r'd) \mod n, \end{cases} \tag{40}$$

where r' is known and k is fixed.

Obviously, we can solve the equations with two unknown and recover the private key d.

B Appendix B

In order to analyze the countermeasures, this appendix introduces the corresponding the verifications of DECDSA and EdDSA, respectively.

B.1 Verification of Deterministic ECDSA

The verification procedure of DECDSA described in Algorithm 3 is same with that of ECDSA.

Algorithm 3 ECDSA Verification

Require: signature results $m, (r, s)$, public key P_A.
Ensure: whether verification succeeds.
1: **if** r **xor** $s \notin [1, n-1]$ **then return false**;
2: $e = SHA(m)$;
3: $u_1 = s^{-1}e \bmod n$, $u_2 = s^{-1}r \bmod n$;
4: $\Omega = u_1 G + u_2 P_A$;
5: **if** $\Omega \neq \mathcal{O}$ **and** $r = x_\Omega \bmod n$ **then return true**;
6: **return false**

B.2 Verification of EdDSA

The verification is shown in Algorithm 4. If some countermeasures are introduced into the calculation of $k = H(d_1, m) \mod n$ during signature, the verification can be successfully passed without any modification.

Algorithm 4 Deterministic EdDSA Verification

Require: message m, signature results (R, s), public key P.
Ensure: Whether the verification succeeds.
1: $r = H(R, P, m) \mod n$;
2: **if** $8sG = 8R + 8rP \in E_{a,d}(F_q)$, **return true**;
3: return **false**;