Lattice-based Fault Attacks against ECMQV

Weiqiong Cao, Hua Chen(✉), Jingyi Feng, Limin Fan and Wenling Wu

Trusted Computing and Information Assurance Laboratory,
Institute of Software, Chinese Academy of Sciences,
South Fourth Street 4#, ZhongGuanCun, Beijing 100190, P.R. China.
{caowq, chenhua, fengjingyi, fanlimin, wwl}@tca.iscas.ac.cn

Abstract. ECMQV is a standardized key agreement protocol based on ECC with an additional implicit signature authentication. In this paper we investigate the vulnerability of ECMQV against fault attacks and propose two efficient lattice-based fault attacks. In our attacks, by inducing a storage fault to the ECC parameter a before the execution of ECMQV, we can construct two kinds of weak curves and successfully pass the public-key validation step in the protocol. Then, by solving ECDLP and using a guess-and-determine method, some information of the victim's temporary private key and the implicit-signature result can be deduced. Based on the retrieved information, we build two new lattice-attack models and recover the upper half of the static private key. Compared with the previous lattice-attack models, our models relax the attack conditions and do not require the exact partial knowledge of the nonces. The validity of the attacks is proven by experimental simulations, which show our attacks pose real threats to the unprotected ECMQV implementations since only one permanent fault is sufficient to retrieve half bits of the secret key.

Keywords: ECC, Fault Attack, Lattice Attack, ECMQV

1 Introduction

Smart cards and mobile devices are playing indispensable roles today, since a lot of important data such as mobile payment data and bank account information is stored on them. Hence, it is necessary to protect their security with cryptographic algorithms. Among various algorithms, elliptic curve cryptosystem (ECC) [1] is a popular one because it ensures the same level of security with less key bits and faster run time than RSA.

It is necessary to analyze not only the theoretical security but also the implementation security resisting physical attacks. When ECC is implemented on embedded devices, physical attacks may gain some information by physical tools to recover the secrets. Among various physical attacks, fault attack(FA) is a powerful one which exploits the faulty results caused by fault injection using laser injection, strong electromagnetic radiation and glitches. So far, many different types of fault attacks(FAs) against ECC have been proposed and weak

curve attack(WCA) based on low-order feature is a common one. In CRYPTO'2000 [2], the WCA based on the faulty basic point was first proposed by Biehl et al. After that, the WCAs based on the faulty curve parameters a and p were also proposed in [3,4]. Differential fault attack(DFA) [2,5,6] is another powerful FA. It recovers the scalar by inducing faults to alter the sign bit or instruction flow during the implementation of a scalar multiplication(SM) kG, and analyzing the difference between the correct and faulty results of the SM. Furthermore, the combination of FA and other attacks has also been used to analyze various algorithms of ECC. In CHES'2011 [7], the combination of FA and simple power analysis(SPA) has been proposed to attack a SM. Besides, FA combining with lattice attack(LA) [6] is also applied to signature algorithms. Nevertheless, to our knowledge, there seems to be no FA on the authenticated key agreement protocol ECMQV.

ECMQV is an extension of MQV proposed by Law, Menezes et al. [8], which has been standardized in IEEE 1363 [9], ANSI X9.63 [10], Chinese standard GM/T 0009-2012 (SM2) [11], etc. It is based on the Diffie-Hellman key agreement protocol on ECC(ECDH) with an additional implicit signature authentication. There mainly exist two kinds of attacks on ECMQV at present. Man-in-the-middle attack based on the application of ECMQV is the first one, such as forgery attack [12] and unknown key-share attack [13]. The other one is the traditional algorithm analysis based on the structure of ECMQV. WCA is naturally the common one and was proposed against one-pass ECMQV in PKC'2003 [14], in which the attacker pretending one party in the agreement sends two low-order points as public keys to the other-party victim. After several runs of the protocol, the victim's private key can be recovered by guessing the implicit-signature results and using Chinese Remainder Theorem(CRT). However, the attack cannot be applied to ECMQV with authentication and public key validation. Meanwhile, Leadbitter and Smart presented a LA against ECMQV in ISC'2003 [15]. If the attacker has partial knowledge of the victim's nonces: the temporary private key and the implicit-signature result, then the LA can be mounted to recover the upper half of the victim's static private key. The remainder bits can be obtained by Baby-Step/Giant-Step algorithm with a run time of $O(n^{1/4})$, where n is the order of basic point in the ECMQV protocol. After that, the combinations of WCA and LA are mentioned in INDOCRYPT'2006 [16] and JMC'2007 [17], respectively. However, such attacks have the following limits: 1) The victim's temporary private key is required to be known to the attacker; 2) There exists no or only part public-key validations to make the victim accept the low-order public keys; 3) The order of the low-order points must have the factor 2^l to ensure l bits of the nonce known for LA, where l is a positive integer. Apparently, the limits above are impractical for an integrated ECMQV. In view of the importance of public-key validation in ECMQV, we think it is interesting if some faults are induced so that the public-key validations are passed. Moreover, because of the existence of the implicit signature, the lattice-based fault attacks against ECDSA probably can be applied to ECMQV.

Our contributions. In this paper, we present two new lattice-based fault attacks against ECMQV. Our attack procedures can be divided into two stages. In the first stage, a storage fault is induced to the ECC parameter a before the running of ECMQV and two kinds of weak curves are constructed. The low-order points on the first weak curve can thereby pass the public-key validation steps in ECMQV. By solving ECDLP and using a guess-and-determine method, some reduced information of the victim's temporary private key r_B and the implicit-signature result s_B can be deduced. In the second stage, we build two new LA models with the retrieved information and successfully recover the upper half of the static private key d_B in ECMQV.

In our attacks, the LA models are more relaxed because it is unnecessary for the attacker to know the partial bits of the nonces s_B and r_B exactly, while it is required in the previous models [15,16,17]. The first model only utilizes the reduced values $s = s_B \mod d$ and $r = r_B \mod d$, where d is the greatest common divisor derived from the weak curves constructed in the first stage. In our case, only when d is equal to 2^l (l is a positive integer), the model is equivalent to the previous model which means the l bits of the nonces have to be known. Except that, the attacker does not need to know any bit of the nonces. The second model is totally different from the previous ones, in which $s = s_B \mod d$ and $r = r_B \mod n_2$ are required. Here d is a small factor of the order of the first weak curve and n_2 is the order of basic point G on the second weak curve. When the sum of bit lengths of n_2 and d is greater than a lower bound, the LA model will work. We also prove the correctness and effectiveness of the two attacks by software simulations. The simulations show that our attacks only require one permanent effective fault to retrieve half bits of the secret key. Thus, the corresponding countermeasure should be considered in practical implementations.

The remainder of the paper is organized as follows: Sect. 2 introduces the ECMQV protocol and some basic theory about lattices. In Sect. 3, the first lattice-based fault attack against ECMQV is presented, and the second one is presented in Sect. 4. The corresponding feasibility is verified by simulations in Sect. 5. Finally, conclusion is given in Sect. 6.

2 Preliminaries

2.1 ECMQV Authenticated Key Agreement Protocol

In this section, we will discuss the elliptic curves in prime field $F_p (p > 3)$ and ECMQV protocol. Elliptic curve $E(a, b)$ is defined by Weierstrass equation

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

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

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

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

Given a basic point $G \in E(F_p)$ with order n, $< G >$ is the group taking G as its generator. For any point $Q \in < G >$, there exists a scalar $k \in [0, n-1]$, so that $Q = kG$. The scalar multiplication(SM) kG is an elementary operation on $E(a, b)$ and is composed of point doublings and additions. There are many algorithms for calculating SM, such as binary algorithm, wNAF window algorithm and Montgomery algorithm [18]. The security of ECC is based on the elliptic curve discrete logarithm problem(ECDLP): knowing the basic point $G \in E(F_p)$ and point $Q \in E(F_p)$, it is hard to find the scalar $k \in [0, n-1]$ satisfying $Q = kG$. As the best general attack on ECDLP, the combination of Pohlig Hellman algorithm and Pollard's rho algorithm reduces the ECDLP in the group $< G >$ into in a subgroup with prime order q, where q is the biggest prime factor of order n and the time complexity is $O(q^{1/2})$. Therefore, the security of ECC depends on the bit-size of q, so the curve parameters and basic point G of ECC are usually selected to make q as big as possible.

Next, we will introduce the three pass **ECMQV** protocol [18]. ECMQV is usually used for negotiating the shared session key between party A and B. In ECMQV, $\#E(F_p)$ is the order of $E(F_p)$, cofactor h is equal to $\#E(F_p)/n$, and $(a, b, p, G, n, \#E(F_p), h)$ are the optional parameters. A and B all have two private-public key pairs, the temporary and the static key pairs, respectively. The temporary key pair is variable with every key agreement. It is assumed that (P_A, d_A), (R_A, r_A) are the static and temporary key pairs of A and (P_B, d_B), (R_B, r_B) are the corresponding key pairs of B, respectively, where $P_A = d_A G$, $R_A = r_A G$, $P_B = d_B G$ and $R_B = r_B G$. In order to resist WCA, it is necessary for both A and B to perform public key validation on each other's static and temporary public keys. As stated in [18], Algorithm 1 is usually used for validating public key.

Algorithm 1 Public Key Validation [18]

Require: parameters a, b, p, n, h and public key Q

Ensure: the validation of Q is pass or not.

1. Verify $Q \neq \mathcal{O}$;
2. Verify that the x/y-values x_Q and y_Q of Q are the elements of field F_p, namely, $x_Q, y_Q \in [0, p-1]$;
3. Verify that Q lies on the elliptic curve $E(a, b)$ defined by a, b and p;
4. If any one of the verifications above fails, then return false; else return true.

Besides, we also define

$$f = \lfloor \log_2 n \rfloor + 1.$$

For any point $Q \in E(F_p)$, let

$$\overline{Q} = x_Q \bmod 2^{\lceil f/2 \rceil} + 2^{\lceil f/2 \rceil}, \tag{3}$$

where x_Q is the x-value of Q, and f is the bit length of n.

There exists an implicit signature with their own static private key to ensure the session key shared by A and B. After that, a key derivation function(KDF) based on hash function is executed to generate the shared key. $KDF(S)$ is the concatenation of the values of hash functions $H(S, i)$, where i is a counter that is accumulated until the sum of the bit lengths of hash values equals the bit length of required key. Meanwhile, as the optional steps, the results are processed by a message authentication code(MAC) algorithm and the result of MAC is sent to the other party for further verification. The whole protocol is specified in Algorithm 2, where ID_A and ID_B are the IDs of A and B, respectively.

Algorithm 2 ECMQV Key Agreement [18]

Require:

$$A \to B : R_A, ID_A;$$
$$B \to A : R_B, ID_B, t_B;$$
$$A \to B : t_A.$$

Ensure: share key K

1. A selects randomly $r_A \in [1, n-1]$, calculates $R_A = r_A G$, and sends R_A, ID_A to B;
2. B calculates the following:
   2.1 validates the public key R_A with Algorithm 1;
   2.2 selects randomly $r_B \in [1, n-1]$ and calculates $R_B = r_B G$;
   2.3 calculates $s_B = r_B + \overline{R_B} d_B \mod n$ and $V = hs_B(R_A + \overline{R_A} P_A)$, and verifies $V \neq \mathcal{O}$;
   2.4 calculates $K = KDF(V, ID_A, ID_B)$;
   2.5 (options)calculates $t_B = MAC(2, V, ID_A, ID_B, R_B, R_A)$;
   2.6 sends R_B, ID_B(, options t_B) to A;
3. A calculates the following:
   3.1 validates the public key R_B with Algorithm 1;
   3.2 calculates $s_A = r_A + \overline{R_A} d_A \mod n$ and $V = hs_A(R_B + \overline{R_B} P_B)$, and verifies $V \neq \mathcal{O}$;
   3.3 calculates $K = KDF(V, ID_A, ID_B)$;
   3.4 (options)calculates $t = MAC(2, V, ID_A, ID_B, R_B, R_A)$, and verifies $t = t_B$;
   3.5 (options) calculates $t_A = MAC(3, V, ID_A, ID_B, R_A, R_B)$, and sends t_A to B;
4. (options) B calculates $t = MAC(3, V, ID_A, ID_B, R_A, R_B)$, verifies $t = t_A$.

2.2 Lattices

In this section, we will introduce some basic definitions of lattice. Lattice is an old mathematical concept. Let vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_d \in \mathbb{R}^m$ are linearly independent,

then the set $\mathcal{L}$

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

is called a lattice and regards the vectors $\boldsymbol{b}_i (i = 1, 2, ..., d)s$ as its basis, where $\mathbb{R}^m$ is the m dimensional space in real number field $\mathbb{R}$. Matrix $B = (\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_d)^T$ is denoted as the basis matrix of $\mathcal{L}$. For any $\boldsymbol{z} \in \mathcal{L}$, there exists $\boldsymbol{x} \in \mathbb{Z}^d$ so that $\boldsymbol{z} = \boldsymbol{x}B$. d is the dimension of $\mathcal{L}$. If $m = d$, then $\mathcal{L}$ is full rank. $\mathcal{L}$ is an integer lattice when any vector $\boldsymbol{b}_i (i = 1, ..., d)$ belongs to $\mathbb{Z}^m$.

There are two famous problems in lattice $\mathcal{L}$, the shortest vector problem(SVP) and the closest vector problem(CVP). For SVP, given the basis $\boldsymbol{b}_i s$ of $\mathcal{L}$, find a nonzero vector $\boldsymbol{v} \in \mathcal{L}$ so that $\|\boldsymbol{v}\| = \lambda_1(\mathcal{L})$, where $\lambda_1(\mathcal{L})$ is the length of shortest vector in lattice $\mathcal{L}$ and $\|.\|$ is denoted as Euclidean norm. It has been proven that LLL algorithm [19] and LLL-based BKZ algorithms [20] can solve approximate SVP in polynomial time. Similarly, CVP is defined as follow: given the 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 $\|\boldsymbol{v} - \boldsymbol{u}\| = \lambda(\mathcal{L}, \boldsymbol{u})$, where $\lambda(\mathcal{L}, \boldsymbol{u})$ is the closest distance from vector $\boldsymbol{u}$ to lattice $\mathcal{L}$. CVP is harder than SVP and the approximate CVP can be solved by using LLL-based Babai's nearest plane algorithm [21] in polynomial time. Hence, CVP is usually reduced into SVP by the embedding technique in practice [22]. Given the basis $\boldsymbol{b}_i s$ of $\mathcal{L}$ and the target vector $\boldsymbol{u}$, a new lattice $\mathcal{L}'$ can be built with new basis $\boldsymbol{b}_1', \boldsymbol{b}_2', ..., \boldsymbol{b}_{d+1}'$, where $\boldsymbol{b}_i' = (\boldsymbol{b}_i, 0)(i = 1, ..., d)$ and $\boldsymbol{b}_{d+1}' = (\boldsymbol{u}, \beta)$. β is a parameter to be determined. If $\boldsymbol{v}$ is the closest vector in $\mathcal{L}$ from $\boldsymbol{u}$, then $(\boldsymbol{u} - \boldsymbol{v}, \beta)$ is the shortest vector in $\mathcal{L}'$.

It has been proved [23] that a full-rank random lattice $\mathcal{L} \in \mathbb{R}^m$ satisfies with overwhelming probability

$$\lambda_1(L) \approx \sqrt{\frac{d}{2\pi e}} vol(\mathcal{L})^{\frac{1}{d}}, \tag{5}$$

where $vol(\mathcal{L})$ is the determinant of $\mathcal{L}$ satisfying $vol(\mathcal{L}) = \prod_{i=1}^{d} \|\boldsymbol{b}_i^*\|$. $\boldsymbol{b}_i^* s$ are the corresponding Gram-Schmidt basis derived from matrix B.

Furthermore, the theorem above can be extended to CVP. Babai has proved [21] that given a target vector $\boldsymbol{u}$, the lattice vector $\boldsymbol{v}$ can be determined in polynomial time when satisfying the inequation

$$\|\boldsymbol{v} - \boldsymbol{u}\| \leq c_1 ||b_N^*||^2 \leq \sqrt{\frac{d}{2\pi e}} vol(\mathcal{L})^{\frac{1}{d}}. \tag{6}$$

3 First Lattice-based Fault Attack against ECMQV

As mentioned above, there exist public key validations described in Algorithm 1 for resisting WCA, and the point V generating shared key cannot be gained directly except the MAC results in Algorithm 2. Therefore, the DFA making

use of the difference between correct and faulty points, and the WCA utilizing the feature of low-order point, all cannot be applied to the ECMQV protocol. However, if we disturb the curve parameter a into a' by fault injection before the execution of ECMQV protocol, then the following public-key validations in ECMQV will be executed on a new weak curve $E_1(a', b)$. Obviously, the low-order points on $E_1(a', b)$ can pass the public key validation. In addition, the basic point $G(x_G, y_G)$ does not lie on the original curve $E(a, b)$ but on another new weak curve $E_2(a', b')$, where $b' = {y_G}^2 - {x_G}^3 - a' x_G$. Thereby, as long as ECMQV protocol can run repeatedly on the two weak curves, we can recover the upper half of the static private key.

In this section, we present the first lattice-based fault attack against ECMQV. To recover the full key, the attack usually composes of three stages. First, FA is carried out to retrieve some reduced information of the nonces. Next, a LA model different from the one in ISC'2003 [15] is built to reveal the upper half of d_B with the retrieved information. Finally, the remaining bits of d_B can be solved by a Baby-Step/Giant-Step algorithm, which is same with the stage presented in ISC'2003 [15] and is not the focus of our study. Hence, our attack just takes the first two stages into account. The following sections describe the FA and its corresponding LA.

3.1 Fault Attack Scenario

Our attack assumes that the attacker as party A intends to acquire the static private d_B of party B and the SM calculation involves the parameter a[1]. Moreover, there exist no additional countermeasures for resisting WCA except the common Algorithm 1. Party A disturbs the parameter a stored in the cryptographic device of party B to generate a faulty a' which is unknown to party A. Meanwhile, the static public key P_A invoked by party B can be changed by party A, which exists in practical applications, such as $P_A = R_A$ in the ECMQV of SSH protocol. Finally, it is assumed that the curve parameter b is quadratic residue, that is, there exists $g \in F_p$ so that $b = g^2 \mod p$. This is true for most of the curves recommended in standards. The point $C(0, \pm g)$ is so-called common point lying on the curve $E(\tilde{a}, b)$ for any $\tilde{a} \in F_p$ as mentioned in [24].

3.2 Fault Attack against ECMQV

The FA includes the following steps, in which fault injection and sending low-order public keys to B are online, and the remaining steps are off-line for analysis.

Step 1: disturb a into a' by fault injection(online). At the beginning of ECMQV in the cryptographic device, the parameter a is written into RAM through the bus. If the attacker mounts FI on the bus/RAM during/after the write operation to disturb a into a', a' will replace a for the following operations of ECMQV and remains unchanged until the device resets or powers down.

[1] a is usually not involved in the SM calculation directly when $a = p - 3$.

Based on the faulty parameter a', we have the first weak curve

$$E_1(a', b) : y^2 = x^3 + a'x + b. \tag{7}$$

n_1 is the order of $E_1(F_p)$.

Meanwhile, because of the faulty a', the SM $R_B = r_B G$ is computed on the second new curve $E_2(a', b')$ instead of the original curve $E(a, b)$.

$$E_2(a', b') : y^2 = x^3 + a'x + b'. \tag{8}$$

n_2 is the order of G on $E_2(a', b')$.

In order to determine the values of a' and b', the attacker first sends the common point $C(0, \pm\sqrt{b})$ lying on $E_1(a', b)$ to B. Obviously, B would accept the point C after validation and send its temporary public key R_B to the attacker. Thereby, the points $G(x_G, y_G)$ and $R_B(x_{R_B}, y_{R_B})$ on the weak curve $E_2(a', b')$ are all known to the attacker. Apparently, a' and b' can be determined by the equations

$$\begin{aligned} {y_G}^2 &= {x_G}^3 + a'x_G + b' \mod p \\ {y_{R_B}}^2 &= {x_{R_B}}^3 + a'x_{R_B} + b' \mod p. \end{aligned} \tag{9}$$

Let $d \in \mathbb{Z}$ be the greatest common divisor of n_1 and n_2, that is, $d = \gcd(n_1, n_2)$, then there exists m_2 so that $n_2 = m_2 d$. To ensure the success of the next LA, we should find an effective faulty a' to make d as big as possible under the feasible time complexity $O(d)$. Otherwise, reset the device and restart FI.

Step 2: send low-order public keys on weak curve $E_1(a', b)$ to B (online). After determining an effective a', the attacker intentionally selects a point R_A lying on $E_1(a', b)$ with order d as its temporary public key and a point P_A satisfying $P_A = uR_A$ as its static public key, where $u \in [1, d-1]$ and $\gcd(d, h + hu\overline{R_A}) = 1$, and then sends them to B. According to Algorithm 2, B calculates the shared key K and outputs R_B ,ID_B, and t_B(options) to the attacker.

Step 3: deduce the reduced information r of the temporary private-key r_B(off-line). Given that $R_B = r_B G$, it follows that $m_2 R_B = r_B(m_2 G)$. Because of the low-order point $m_2 G$, it is easy to solve the ECDLP and gain the result $r = r_B \mod d$, i.e., $r_B = r + \lambda d$, where $\lambda < n/d$.

Step 4: guess and determine the reduced information s of the implicit-signature result s_B(off-line). Since $P_A = uR_A$ and $\gcd(d, h+hu\overline{R_A}) = 1$, $h(R_A + \overline{R_A}P_A)$ lies on $E_1(a', b)$ and its order $\frac{d}{\gcd(d, h+hu\overline{R_A})}$ equals d. Guess the reduced value $s = s_B \mod d$ and calculate the following formulas

$$\begin{aligned} V &= hs(R_A + \overline{R_A}P_A), \\ K &= KDF(V, ID_A, ID_B), \\ t &= MAC(2, V, ID_A, ID_B, R_B, R_A). \end{aligned} \tag{10}$$

As long as $t = t_B$, the corresponding guessed s is the correct value and $s_B = s + \mu d$, where $\mu < n/d$.

As an option in Algorithm 2, B may terminate the ECMQV agreement without calculating t_B. In that case, the attacker needs to implement the encryption/decryption using the shared key K with B. If the results of encryption/decryption are correct, the guessed s is also correct. Besides, in case the static public key P_A is sent to B before FI and cannot be changed by the attacker, the attacker could construct a low-order point $R_A + \overline{R_A}P_A$ with order n_3 on a new weak curve by uprating R_A. d will become the greatest common divisor of n_2 and n_3 for analysis by then.

To sum up, by the fault attack above, the attacker can get some reduced information of r_B and s_B, i.e., r and s, which can be applied to build the model of lattice attack.

3.3 Lattice Attack against ECMQV

As stated above, although the attacker does not know the exact partial bits of r_B and s_B as presented in ISC'2003 [15], the LA still can be mounted with the reduced information retrieved by FA.

Assuming that the ECMQV protocol based on the faulty parameter a' is executed N times, the attacker gets N reduced results (r_i, s_i) by FA. For $i = 1, \ldots, N$, the i-th temporary private key $r_{B,i}$ and the i-th implicit-signature result $s_{B,i}$ satisfy the following equations, respectively.

$$\begin{aligned} r_{B,i} &= r_i + \lambda_i d, \\ s_{B,i} &= s_i + \mu_i d. \end{aligned} \tag{11}$$

Where $\lambda_i, \mu_i < n/d$.

As shown in Algorithm 2, it is known

$$s_{B,i} = r_{B,i} + \overline{R_{B,i}} d_B \mod n, \tag{12}$$

where $R_{B,i}$ is the i-th temporary public key, and $\overline{R_{B,i}} \in [2^{\lceil f/2 \rceil}, 2^{\lceil f/2 \rceil + 1} - 1]$ is derived from the equation (3).

Substituting the equations (11) into (12), we have

$$(\mu_i - \lambda_i)d = r_i - s_i + \overline{R_{B,i}} d_B \mod n. \tag{13}$$

Hence, there exists $h_i \in \mathbb{Z}$ satisfying the equation

$$(\mu_i - \lambda_i) = (d^{-1}\overline{R_{B,i}} \mod n)d_B + h_i n - d^{-1}(s_i - r_i) \mod n. \tag{14}$$

Since $\lambda_i, \mu_i < n/d$, we have

$$|h_i n + (d^{-1}\overline{R_{B,i}} \mod n)d_B - d^{-1}(s_i - r_i) \mod n| < n/d. \tag{15}$$

A model of LA can be built by the inequation (15). Let $A_i = d^{-1}(s_i - r_i) \mod n$, $B_i = d^{-1}\overline{R_{B,i}} \mod n$. For $i = 1, \ldots, N$, a lattice $\mathcal{L}$ can be spanned by the row

vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{N+1}$ of matrix

$$M = \begin{pmatrix} n & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & n & 0 \\ B_1 & \cdots & B_N & 1/d \end{pmatrix}.$$

Let $\boldsymbol{x} = (h_1, \ldots, h_N, d_B) \in \mathbb{Z}^{N+1}$, then $\boldsymbol{x}M$ is a nonzero vector in $\mathcal{L}$ and $\boldsymbol{v} = \boldsymbol{x}M = (B_1 d_B + h_1 n, \ldots, B_N d_B + h_N n, d_B/d)$. In addition, let the non-lattice vector $\boldsymbol{u} = (A_1, \ldots, A_N, 0) \in \mathbb{Z}^{N+1}$. Naturally, the inequation (15) can be rewritten into

$$\|\boldsymbol{v} - \boldsymbol{u}\| < \sqrt{N+1}n/d \tag{16}$$

As mentioned in Sect. 2, if $\sqrt{N+1}n/d < \sqrt{\frac{N+1}{2\pi e}} vol(\mathcal{L})^{\frac{1}{N+1}}$, i.e., $N > \frac{f + \log 2\pi e}{l_d - \log 2\pi e}$, then $\boldsymbol{v}$ can be determined by solving CVP, where $vol(\mathcal{L}) = det(M) = n^N/d$ and $l_d = \lceil \log d \rceil$ is the bit length of d. Nevertheless, due to $\overline{R_{B,i}} \in [2^{\lceil f/2 \rceil}, 2^{\lceil f/2 \rceil + 1} - 1]$, there is only the upper half of d_A recovered in the lattice attack as proved in ISC'2003 [15].

4 Second Lattice-based Fault Attack against ECMQV

In this section, we will introduce the second lattice-based FA against ECMQV. The target of fault injection is still the parameter a and the FA scenario is same with the first attack. However, the two constructed weak curves and the model of LA have some different features.

4.1 Fault Attack against ECMQV

Similarly, the steps of fault attack are mainly divided into two parts, online and off-line.

The online steps:

As stated above, after disturbing a into a' by fault injection repeatedly, we obtain the two weak curves $E_1(a', b)$ and $E_2(a', b')$. But unlike the first attack, it assumes that the order n_1 of $E_1(a', b)$ have a small factor d and the ECDLP on $E_2(a', b')$ is solvable, that is, the time complexity $O(d)$ and $O(\sqrt{q})$ are feasible for calculation, where q is the greatest prime factor of order n_2 of the basic point G on $E_2(a', b')$.

Next, using the same method as the first attack, the attacker selects the low-order point R_A and P_A on $E_1(a', b)$ as its public keys and sends them to B, where the selected R_A and P_A are same as those in the first attack. Finally, the attacker receives the corresponding R_B ,ID_B, and t_B(options) from B.

The off-line analysis steps:

First, the reduced information of temporary private key r_B is deduce by solving ECDLP. Given that $R_B = r_B G$ and the time complexity $O(\sqrt{q})$ is feasible,

we can deduce the value $r \in [1, n_2 - 1]$ by using Pohlig-Hellman algorithm and Pollard's rho algorithm, so $r_B = r + \lambda n_2$, where $\lambda \in \mathbb{Z}$ and $\lambda < n/n_2$.

Next, the correct value $s \in d$ is determined by using the same guess-and-determine method as the first attack, and then $s_B = s + \mu d$, where $\mu < n/d$.

Although it is uncertainty whether there exists an available common divisor between n_2 and d, a model of lattice attack still can be built.

4.2 Lattice Attack against ECMQV

In the same way, after the faulty ECMQV runs N times, we have the following equations for $i = 1, \ldots, N$.

$$\begin{aligned} r_{B,i} &= r_i + \lambda_i n_2, \\ s_{B,i} &= s_i + \mu_i d. \end{aligned} \tag{17}$$

Where $\lambda_i, \mu_i \in \mathbb{Z}$, $\lambda_i < n/n_2$ and $\mu_i < n/d$.

Substitute the equations (17) into the equation $s_{B,i} = r_{B,i} + \overline{R_{B,i}} d_B \bmod n$, then

$$\begin{aligned} s_i + \mu_i d &= r_i + \lambda_i n_2 + \overline{R_{B,i}} d_B \bmod n (i > 1) \\ s_1 + \mu_1 d &= r_1 + \lambda_1 n_2 + \overline{R_{B,1}} d_B \bmod n (i = 1). \end{aligned} \tag{18}$$

We have the following $N - 1$ equations by eliminating d_B.

$$\begin{aligned} \mu_i = {} & d^{-1}(r_i - s_i) - d^{-1}\overline{R_{B,1}}^{-1}\overline{R_{B,i}}(r_1 - s_1) - d^{-1}\overline{R_{B,1}}^{-1}\overline{R_{B,i}} n_2 \lambda_1 \\ & + d^{-1} n_2 \lambda_i + \overline{R_{B,1}}^{-1}\overline{R_{B,i}} \mu_1 \mod n (1 < i \le N) \end{aligned} \tag{19}$$

Let $A_i = d^{-1}\overline{R_{B,1}}^{-1}\overline{R_{B,i}}(r_1 - s_1) - d^{-1}(r_i - s_i) \bmod n$, $B_i = -d^{-1}\overline{R_{B,1}}^{-1}\overline{R_{B,i}} n_2 \bmod n$, $C = d^{-1} n_2 \bmod n$ and $D_i = \overline{R_{B,1}}^{-1}\overline{R_{B,i}} \bmod n$, then there exists $h_i \in \mathbb{Z}$ so that

$$\mu_i = B_i \lambda_1 + C \lambda_i + D_i \mu_1 + h_i n - A_i. \tag{20}$$

Since $\mu_i < n/d$, we have

$$|B_i \lambda_1 + C \lambda_i + D_i \mu_1 + h_i n - A_i| < n/(2d). \tag{21}$$

Similarly, for $i = 2, \ldots, N$, we can construct a lattice $\mathcal{L}$ spanned by the row vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2N}$ of matrix

$$M = \begin{pmatrix} \delta n & & & \cdots & & 0 \\ 0 & \ddots & & & & \\ & & \delta n & & & \\ \delta D_2 & \cdots & \delta D_N & \delta & & \vdots \\ \delta B_2 & \cdots & \delta B_N & 0 & \gamma & \\ \delta C & & & & \ddots & \\ & \ddots & & & & \gamma \\ 0 & & \delta C & 0 & \cdots & \gamma \end{pmatrix},$$

where $\gamma, \delta \in \mathbb{R}$. Let $\boldsymbol{x} = (h_2, \ldots, h_N, \mu_1, \lambda_1, \ldots, \lambda_N) \in \mathbb{Z}^{2N}$, then $\boldsymbol{v} = \boldsymbol{x}M = (\delta(h_2 n + D_2\mu_1 + B_2\lambda_1 + C\lambda_2), \ldots, \delta(h_N n + D_N\mu_1 + B_N\lambda_1 + C\lambda_N), \delta\mu_1, \gamma\lambda_1, \ldots, \gamma\lambda_N)$. In addition, let non-lattice vector $\boldsymbol{u} = (\delta A_2, \ldots, \delta A_N, 0, \cdots, 0) \in \mathbb{Z}^{2N}$, then $\boldsymbol{v} - \boldsymbol{u} = (\delta\mu_2, \ldots, \delta\mu_N, \delta\mu_1, \gamma\lambda_1, \ldots, \gamma\lambda_N)$. Supposing that $\beta \in \mathbb{R}, \delta = d\beta$ and $\gamma = n_2\beta$, we have

$$\|\boldsymbol{v} - \boldsymbol{u}\| < \sqrt{2N} n\beta \tag{22}$$

It is assumed that $l_2 = \lceil \log n_2 \rceil$ and $l_d = \lceil \log d \rceil$. If $l_2 + l_d > f + \log 2\pi e$ and $N > f/(l_2 + l_d - f - \log 2\pi e)$, then $\|\boldsymbol{v} - \boldsymbol{u}\| < \sqrt{2N} n\beta < \sqrt{\frac{2N}{2\pi e}} vol(L)^{\frac{1}{2N}}$, where $vol(L) = \det(M) = \beta^{2N} n^{N-1} d^N {n_2}^N$. Hence, $\boldsymbol{v}$ can be determined by solving CVP, and then $d_B = \overline{R_{B,1}}^{-1}(s_1 - r_1 + \delta\mu_1/\beta - \gamma\lambda_1/\beta) \bmod n$. Similarly, since $\overline{R_{B,i}}$ belongs to $[2^{\lceil f/2 \rceil}, 2^{\lceil f/2 \rceil + 1} - 1]$, only the upper half of d_A can be recovered by the LA.

5 Feasibility Analysis and Simulation of Attacks

In this section, we will verify the feasibility of our proposed attacks by software simulations. First, we carry out some simulations in two standard curves to analyze the rate of effective faulty parameter $a's$ for the attacks. Next, based on the effective faulty $a's$, we verify the two attacks by experiments.

We choose the curves in prime field with 256-bit keys recommended in FIPS 186-2 and SM2 as the FI objects, and then simulate the flipped single fault and 32-bit random fault of parameter a, respectively. For the flipped single fault, every bit of a is flipped in turn, so there are 256 kinds of different faulty $a's$. As for the random fault, a continuous 32-bit part of a is randomized, which is also simulated 256 times. After that, we compute the orders n_1, n_2 of the two constructed curves by using the MIRACL implementation of SEA algorithm [25], respectively.

As stated above in the first attack, the number N of ECMQV protocol needed for LA is greater than $\frac{f + \log 2\pi e}{l_d - \log 2\pi e}$. Hence, the case $l_d \geq 5 (i.e., \lceil \log d \rceil > \lfloor \log 2\pi e \rfloor)$ is required for the two weak curves. Moreover, the greater d, the smaller N. Fig. 1 displays the cumulative outcome probability that the greatest common divisor d is bigger than a certain bit length for the two faulty types of the two standard curves. To better understand these results, we list the faulty number N_d, d_{max}(namely the biggest d) and the bit length $\lceil \log d_{max} \rceil$ of d_{max} when $l_d \geq 5$ as shown in Table 1. From the results the probability of faulty $a's$ available for the first attack exceeds 4%, and the optimal $d_{max}s$ for the four faulty types are sufficient to mount lattice attack successfully.

As for the second attack, our analysis concerns the probability that the ECDLP with time complexity $O(2^{l_q/2})$ is solved by modern computers, where $l_q = \lceil \log q \rceil$ and q is the biggest prime factor of order n_2 on $E_2(a', b')$. We assume that the computation limit for solving ECDLP is bound to 112 bits complexity [26], thus we consider the faulty $a's$ whose q is smaller than 112 bits are effective for the attack. Meanwhile, in order to ensure the success rate of LA, $l_2 + l_d > f + \log 2\pi e$ is also required under the premise of feasible computation

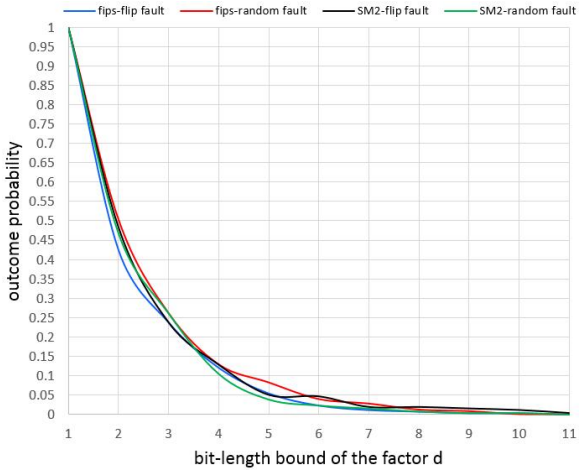

Fig. 1. Cumulative probability of the bit length of each common factor d in the first attack

Table 1. Effective faulty a's for the first attack

data / curve	N_d	$N_d/256$	d_{max}	$\lceil \log d_{max} \rceil$
FIPS-flipped fault	14	5.5%	0x374	10
FIPS-random fault	21	8.2%	0x1E9	9
SM2-flipped fault	13	5.1%	0x409	11
SM2-random fault	10	4.0%	0x2B0	10

complexity $O(2^{l_d})$. Similarly, Fig. 2 displays the cumulative outcome probability that the biggest factor q is smaller than a certain bit-length. It can be observed that the probability to obtain sufficiently small sizes of q exceeds 19%. Table 2 lists the number N_{l_q} of all the faulty $a's$ under the conditions $l_q \leq 112$, $l_d < 40$ and $l_2 + l_d > f + \log 2\pi e$, in which q_{min} is the smallest q satisfying the conditions above. L equals the biggest $l_2 + l_d - f - \lfloor \log 2\pi e \rfloor$ when $q = q_{min}$. Compared to the first attack, the probability $N_{l_q}/256$ of effective faulty $a's$ in the second attack is greater than $N_d/256$, but at the same time the computation complexity is higher as well.

To sum up, there are at least 24% faulty $a's$ available for both of the above attacks in all. Which attack to choose depends on the case generated by the faulty a'.

Finally, we carry out the two attacks based on the flipped single fault of the FIPS 186-2 curve, in which the lattice attacks invoke the Babai algorithm based on BKZ reduced basis in NTL library [27]. In the first LA, the upper half of d_A can be recovered correctly by LA as long as $l_d \geq 5$, and at least $N = 62$ is needed when $l_d = 5(l_d - \lfloor \log 2\pi e \rfloor = 1)$. The N needed is far smaller

Fig. 2. Cumulative probability of the bit length of each biggest factor q in the second attack

Table 2. Effective faulty a's for the second attack

data / fault type	N_{l_q}	$N_{l_q}/256$	$q = q_{min}$			
			$\lceil \log q_{min} \rceil$	l_2	l_d	L
FIPS-flipped fault	56	21.9%	58	253	8	1
FIPS-random fault	43	16.8%	55	256	39	35
SM2-flipped fault	49	19.1%	60	255	39	26
SM2-random fault	60	23.4%	57	256	39	35

than the theoretical one(i.e., $N > \frac{f+\lfloor \log 2\pi e \rfloor}{l_d - \lfloor \log 2\pi e \rfloor} = 260$). Meanwhile, we choose the faulty case $\lceil \log q \rceil = 58$, $l_2 = 253$, $l_d = 8$ and $L = 1$ for the second attack. The experiments show that at least $N = 90$ is needed for a successful LA when $l_d + l_2 - f - \lfloor \log 2\pi e \rfloor = 1$. This is also far smaller than $\frac{f}{l_d + l_2 - f - \lfloor \log 2\pi e \rfloor} = 256$. Thus, the results from lattice attack in practice are actually better than those in theory.

6 Conclusion

In this paper, we present two new lattice-based fault attacks on ECMQV, which are based on the possibility of storage error of parameter a. Both of the attacks construct two weak curves with the faulty parameter a'. Because of the greatest common divisor d of the two curves, the first attack reduces the temporary private key and implicit-signature result (r_B, s_B) into $(r_B \mod d, s_B \mod d)$, respectively. The second attack reduces (r_B, s_B) into $(r_B \mod n_2, s_B \mod d)$ by solving ECDLP on the second weak curve $E_2(a', b')$ and using the guess-determine method. Next, the two new lattice attacks with the reduced

information of (r_B, s_B) are launched to recover half bits of the static private key d_B. Finally, the experiments confirm the feasibility of our attacks. For a 256-bit standard key length, 62 faulty agreements with a 5 bit-length common divisor d are sufficient to recover 128 bits of the private key d_B in the first attack, and 90 faulty agreements are sufficient to determine half of d_B using the second attack when the sum of the bit lengths of the small factor d and order n_2 equals 261.

The ideas of such attacks also can be applied to the other ECC algorithms, such as ECDSA and SM2 signature. Note that although the point R_A sent to victim from attacker can pass through the public key validation, the general countermeasure such as the point validation toward the points G, Q_B during the calculation of SM $Q_B = r_B G$ is effective at resisting our attacks. For this reason, our further research will focus on how to mount attacks when there are some countermeasures in SM. For example, we can consider fault attacks based on the storage error of parameter p.

Acknowledgments. We thank the anonymous reviewers for their careful reading and insightful comments. This work is supported by China's National Cryptography Development Fund (No.MMJJ20170214 and No.MMJJ20170211), National Natural Science Foundation (No.61672509) and National Science and Technology Major Project (No.2014ZX01032401-001).

References

1. Miller, V.: Use of Elliptic Curves in Cryptography. In: Advances in Cryptology-CRYPTO'85 Proceedings, Springer (1986) 417–426
2. Biehl, I., Meyer, B., Müller, V.: Differential Fault Attacks on Elliptic Curve Cryptosystems. In: Advances in Cryptology-CRYPTO 2000, Springer (2000) 131–146
3. Ciet, M., Joye, M.: Elliptic curve cryptosystems in the presence of permanent and transient faults. Designs Codes Cryptography **36**(1) (2005) 33–43
4. Kim, T., Tibouchi, M.: Bit-flip faults on elliptic curve base fields, revisited. In: International Conference on Applied Cryptography and Network Security. (2014) 163–180
5. Blömer, J., Otto, M., Seifert, J.P.: Sign Change Fault attacks on Elliptic Curve Cryptosystems. In: Fault Diagnosis and Tolerance in Cryptography. Springer (2006) 36–52
6. Schmidt, J., Medwed, M.: A Fault Attack on ECDSA. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2009 Workshop on, IEEE (2009) 93–99
7. Fan, J., Gierlichs, B., Vercauteren, F.: To infinity and beyond: Combined attack on ecc using points of low order. In: International Workshop on Cryptographic Hardware and Embedded Systems. (2011) 143–159
8. Elkamchouchi, H.M., Abu Elkair, E.F.: An efficient protocol for authenticated key agreement. Designs Codes Cryptography **28**(2) (2003) 119–134
9. Std, I.: 1363-2000 - ieee standard specifications for public-key cryptography. IEEE Computer Society August (2000) 1–228
10. Association, A.B.: Public key cryptography for the financial services industry, key agreement and key transport using elliptic curve cryptography. Speculum **81**(2) (2006) 566–569

11. of State Commercial Cryptgraphy Administration, O.: Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves(in Chinese). http://www.oscca.gov.cn/UpFile/2010122214822692.pdf (2010)
12. Yeh, H.T., Sun, H.M., Hwang, T.: Improved authenticated multiple-key agreement protocol. Computers, Mathematics with Applications **46**(2) (2003) 207–211
13. Kaliski, B.S.: An unknown key-share attack on the mqv key agreement protocol. Acm Transactions on Information System Security **4**(3) (2001) 275–288
14. Antipa, A., Brown, D., Menezes, A., Struik, R., Vanstone, S.: Validation of elliptic curve public keys. In: International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography. (2003) 211–223
15. Leadbitter, P.J., Smart, N.P.: Analysis of the insecurity of ecmqv with partially known nonces. In: International Conference on Information Security(ISC). (2003) 240–251
16. Menezes, A., Ustaoglu, B.: On the Importance of Public-Key Validation in the MQV and HMQV Key Agreement Protocols. Springer Berlin Heidelberg (2006)
17. Menezes, A.: Another look at hmqv. JMC **1**(1) (2007) 47–64
18. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to elliptic curve cryptography. Springer (2004)
19. Lenstra, H.W., Lenstra, A.K., Lovfiasz, L.: Factoring polynomials with rational coeficients. In: Mathematische Ann. (1982) 515–534
20. Schnorr, C.P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theoretical Computer Science **53**(2-3) (1987) 201–224
21. Babai, L.: On lovász' lattice reduction and the nearest lattice point problem (shortened version). Combinatorica **6**(1) (1986) 1–13
22. Nguyen, P.Q., Stern, J.: Lattice reduction in cryptology: An update. Lecture Notes in Computer Science **1838** (2000) 85–112
23. M, A.: Generating random lattices according to the invariant distribution. Draft of March (2006)
24. Battistello, A. In: Common Points on Elliptic Curves: The Achilles' Heel of Fault Attack Countermeasures. Springer International Publishing, Cham (2014) 69–81
25. Schoof, R.: Counting points on elliptic curves over finite fields. Journal de Theorie des Nombres de Bordeaux **7**(1) (1995) 219–254
26. Bos, J.W., Kaihara, M.E., Kleinjung, T., Lenstra, A.K., Montgomery, P.L.: Solving a 112-bit prime elliptic curve discrete logarithm problem on game consoles using sloppy reduction. International Journal of Applied Cryptography **2**(3) (2012) 212–228
27. Shoup, V.: Number Theory C++ Library (NTL) version 9.6.4. http://www.shoup.net/ntl/ (2016)