SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2018, Vol. 61 032105:1–032105:14
doi: 10.1007/s11432-016-9030-0

Attacking OpenSSL ECDSA with a small amount of side-channel information

Wenbo WANG[1,2] & Shuqin FAN[2*]

[1]*Luoyang University of Foreign Languages, Luoyang* 471003*, China;*
[2]*State Key Laboratory of Cryptology, Beijing* 100878*, China*

Received 25 August 2016/Revised 13 December 2016/Accepted 21 January 2017/Published online 30 August 2017

Abstract In this work, we mount a lattice attack on the ECDSA signatures implemented by the latest version of OpenSSL which uses the windowed non-adjacent form method to implement the scalar multiplication. We first develop a new way of extracting information from the side-channel results of the ECDSA signatures. Just given a small fraction of the information about a side-channel result denoted as double-and-add chain, we take advantage of the length of the chain together with positions of two non-zero digits to recover information about the ephemeral key. Combining the information of both the most significant digits and the least significant bits, we are able to gain more information about the ephemeral key. The problem of recovering ECDSA secret key is then translated to the hidden number problem which can be solved by lattice reduction algorithms. Our attack is mounted to the **secp256k1** curve, and the result shows that 85 signatures would be enough to recover the secret key, which is better than the result that previous attack gained only utilizing the information extracted from the least significant bits, using about 200 signatures to recover the secret key.

Keywords ECDSA, OpenSSL, lattice attack, windowed non-adjacent form, hidden number problem, Flush+Reload attack

Citation Wang W B, Fan S Q. Attacking OpenSSL ECDSA with a small amount of side-channel information. Sci China Inf Sci, 2018, 61(3): 032105, doi: 10.1007/s11432-016-9030-0

1 Introduction

Digital signature schemes can be used to provide the data integrity, data origin authentication and non-repudiation for basic cryptographic services. They are widely used as primitives in cryptographic protocols that support entity authentication, authenticated key transport and authenticated key agreement [1–3]. As a popular used algorithm, the digital signature algorithm (DSA) [4,5] was specified in the FIPS, known as the digital signature standard (DSS). The elliptic curve digital signature algorithm (ECDSA) [6], which was first proposed by Vanstone [7] in 1992, is the elliptic curve variant of the DSA. It has been widely used in many situations, including the Austrian Citizen Card, the Apple's CommonCrypto framework (as included in iOS versions 7.1.2 through 8.3) and the Bitcoin [8], etc.

OpenSSL implementation of ECDSA and its possible side-channel attacks. One of the most popular software implementations of ECDSA is its OpenSSL [9] implementation. As a commonly used open-source cryptographic library, OpenSSL has been widely used to implement many cryptographic protocols and standards, including the secure sockets layer (SSL) protocol, transport layer security (TLS)

* Corresponding author (email: fansq@sklc.org)

protocol, the OpenPGP standard, etc. For elliptic curves over characteristic two fields, the Montgomery ladder method is used in OpenSSL to implement the scalar multiplication, which has been attacked by Yarom and Benger [10] using the side-channel attack with one signature fully recovering the ECDSA private key. For elliptic curves over prime fields, the windowed non-adjacent form (wNAF) method is used to implement the scalar multiplication, which is now being used by default in the OpenSSL implementation of ECDSA in its latest version (version 1.0.2h, published in May 3, 2016). We only focus on those curves over prime fields with wNAF methods in this paper.

In a real-world scenario, implementation issues are always the main vulnerabilities that influence the security of the algorithm. After the initial work of side-channel attack by Kocher et al. [11], more side-channel attacks [12–18] have been well developed to mount on software implementations. In the implementation of the ECDSA algorithm, one core operation is to implement the scalar multiplication of the given point G on the elliptic curve by the ephemeral key k. One recent work of the cache side-channel attack that named the Flush+Reload attack is proposed by Yarom and Falkner [19] in 2014, which can get information about k by observing the execution of kG. It can get the sequence of point additions and doublings (denoted as the double-and-add chain) which are used to implement the scalar multiplication. More recently, Allan et al. [20] proposed an amplified side-channel attack, being able to extract more information. Their new side-channel attack manages to obtain a perfect double-and-add chain without error with probability being nearly 100%. Given the double-and-add chain, one can seek to retrieve information about the ephemeral key, thus being able to recover the secret key. For different methods of implementing the scalar multiplication, the information that can be obtained from the double-and-add chain is different.

There are several ways to implement the scalar multiplication, among which the following three methods are perhaps the most popular ones: double-and-add method, the sliding-window method and the wNAF method. Suppose we get the double-and-add chain of the scalar multiplication by Flush+Reload attack. If the double-and-add method is used for scalar multiplication, then getting the double-and-add chain indicates that the scalar itself be fully recovered; if the sliding-window method is used, then applying the method described in [21], the secret key can be easily recovered since the double-and-add chain helps to directly determine some discrete bits of the ephemeral key. While for the wNAF method, recovering the secret key would not be that easy, since it is not convenient to obtain direct information on bits of scalar from the double-and-add chain. In fact, one cannot get any direct information of the scalar except for several least significant bits (LSBs) [22].

Attacks against ECDSA. It has been known that the leakage of the ephemeral key can be used to recover the private key of ECDSA. It was first in 2001 that Howgrave-Graham and Smart [23] proposed their attack against DSA heuristically under the assumption that some consecutive leaked ephemeral key bits be known by side-channel attacks. In 2002, Nguyen and Shparlinski [24] analyzed their method in further details and gave a provable polynomial-time attack against DSA when some consecutive bits (e.g., the LSBs) of the ephemeral keys were leaked. They also extended their results to ECDSA [25] and managed to recover a 160-bit private key with 100 signatures each leaking 3 consecutive least significant bits. The basic idea is to reduce the key recovery problem to an instance of the hidden number problem (HNP), which can be further reduced to the closest vector problem (CVP) in a suitable lattice, with the knowledge of consecutive leaked ephemeral key bits. The best result of this method so far is achieved by Liu and Nguyen [26] in 2013, using the algorithm BKZ 2.0 [27], which is one of the best lattice reduction algorithms up to now. A 160-bit signature with 2-bit leakage of LSBs can be recovered with 100 signatures.

The attacks mentioned above are mainly based on the fact that at least several bits of the ephemeral key k are definitely known. However, this would become quite hard when the wNAF method is used in implementation of ECDSA as the former one does not reveal direct information on bits of the ephemeral key.

Benger et al. [22] proposed a method to extract the LSBs of the ephemeral key with the knowledge of double-and-add chain of the execution of scalar multiplication in ECDSA implementation via the Flush+Reload attack, being able to recover the private key of the **secp256k1** curve using about 200

signatures with success probability being 3.5%. However, the number of required signatures is too big, as the average number of leaked LSBs is about $\sum_{i=0}^{\infty} i/2^i \approx 2$ per signature.

In [28], van de Pol et al. proposed a new way of obtaining information about the ephemeral key by extracting information indirectly from a perfect double-and-add chain. They managed to obtain 47.6 bits of information per signature on average, which is far more than the result of Benger et al., leading to 13 signatures recovering the private key of the **secp256k1** curve based on results of the Flush+Reload attack. However, this method requires that the double-and-add chain be perfect, which means that it should be full and without any error. Meanwhile it relies on the property of some special curves, i.e., the order q of point G can be expressed as $2^n - \varepsilon$, where $|\varepsilon| < 2^p$, $p \approx n/2$. On the other hand, the instances used to construct the HNP are not relatively independent, with the information of one signature being used more than once, which means that the correct solution might not correspond to the lattice problem.

In [29], Fan et al. proposed a new effective way of extracting and utilizing information from the perfect double-and-add chain, being able to obtain as much as 105.8 bits of information per signature on average for 256-bit ECDSA based on the result of the Flush+Reload attack. The problem of recovering the secret key is then translated to the extended hidden number problem (EHNP) which can be solved by lattice reduction algorithm. They managed to successfully recover the secret key of the **secp256k1** curve with only 4 signatures. Their method does not rely on the special property of q, and each signature is only used once to construct an EHNP instance, which fully avoids the drawbacks in [28].

Our contribution. In this paper, we give a lattice attack on the ECDSA implementation in the latest version of OpenSSL which uses the wNAF method to implement the scalar multiplication, using only a small fraction of information of the double-and-add chain of the ephemeral key. We propose a new way of extracting information about the ephemeral key, only using the length of the double-and-add chain together with positions of the second non-zero digit and the last non-zero digit (counted from the higher index). Combining the information of both the most significant digits and the LSBs, we are able to gain no less than 2.99 bits of information on average per signature on 256-bit curve. The problem of recovering private key is then translated to the hidden number problem, which can be solved by lattice reduction algorithms. Our attack is mounted to the **secp256k1** curve, and the result shows that 85 signatures would be enough to recover the secret key.

Comparing with [29], our attack only uses a small fraction of the double-and-add chain of the ephemeral key k. In fact, we only need to know the positions of the second non-zero digit and the last non-zero digit (from the higher index) together with the length of the chain rather than obtaining a perfect one from the side-channel attack. We utilize only the most significant bits together with the least significant bits rather than utilizing all positions of the double-and-add chain. From this aspect, our attack may be more convenient to be used. On the other hand, we are able to obtain averagely no less than 2.99 bits information about the ephemeral key each signature, and even about 3.99 bits on average for $q \approx 2^n - \epsilon$ where $\epsilon > 0$ is small, which is more than the 2 bits on average in [22] extracting information only from the least significant bits. So reasonably the required signatures are decreased from 200 to 85. We also propose some selection rules on signatures to improve the attack results.

2 Preliminaries

In this section we briefly recall the elliptic curve digital signature algorithm (ECDSA). We further introduce its implementation in OpenSSL which use the wNAF representation to implement scalar multiplication and the possible attack against it. After that the Hidden Number problem is introduced.

2.1 The elliptic curve digital signature algorithm (ECDSA)

Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$ where p is prime and $G \in E$ be a fixed point of a large prime order q. Both G and q are publicly known. The private key of a signer is an integer $0 < \alpha < q$, and the public key is the point $Q = \alpha G$. Given a hash function H, the ECDSA signature (r, s) of a message m is computed as follows:

(1) Choose an ephemeral key k randomly such that $0 < k < q$.
(2) Compute the point $(x, y) = kG$, and let $r \equiv x \bmod q$; if $r = 0$, go back to the first step.
(3) Compute $s = k^{-1}(H(m) + r \cdot \alpha) \bmod q$; if $s = 0$, go back to the first step.

Given the knowledge of the ephemeral key k and (s, r, m), the private key can be easily recovered by

$$\alpha = r^{-1}(s \cdot k - H(m)) \mod q. \tag{1}$$

2.2 OpenSSL implementation of ECDSA by wNAF and its possible attack

Let us first describe the basic implementation of scalar multiplication using the windowed Non-adjacent Form (wNAF) representation, which is used by default in the latest version of OpenSSL.

First, a window size w is chosen (e.g., for curve **secp256k1**, $w = 3$) before computing the scalar multiplication kG. Then precomputation and storage of the points $\{\pm G, \pm 3G, \ldots, \pm(2^w - 1)G\}$ are executed. After that the scalar k is converted to its non-adjacent form (NAF). Algorithm 1 introduces the concrete method for converting a scalar into its NAF. Suppose there are totally l non-zero digits in the output sequence $\{e_i\}$ of Algorithm 1. Denote the i-th non-zero digit in $\{e_i\}$ as k_i, where $k_i \in \{\pm 1, \pm 3, \ldots, \pm(2^w - 1)\}$ for $1 \leqslant i \leqslant l$. Let λ_i be the position of each k_i in the chain of $\{e_i\}$, then for $i \geqslant 2$, $\lambda_i - \lambda_{i-1} \geqslant w + 1$ and the scalar k can be rewritten as $k = \sum_{i=1}^{l} k_i \cdot 2^{\lambda_i}$. We call k_l the most significant digit (MSD), and MSDs denotes k_l together with k_{l-1}.

Algorithm 1 Conversion to wNAF form

Require: Scalar k and window size w;
Ensure: e_0, e_1, e_2, …, e_{λ_l} is the wNAF form of k;

```
i ⇐ 0;
while k > 0 do
    if k mod 2 = 1 then
        e_i = k mod 2^{w+1};
        if e_i >= 2^w then
            e_i ⇐ e_i − 2^{w+1};
        end if
        k ⇐ k − e_i;
    else
        e_i ⇐ 0;
    end if
    k ⇐ k/2;
    i ⇐ i + 1;
end while
```

After converting k to the wNAF form, the multiplication kG is executed as the Algorithm 2 describes. In the actual OpenSSL execution, the bitlength of k is set to a fixed value of $\lfloor \log_2 q \rfloor + 1$ by adding q or $2q$ to the ephemeral key, which can resist the Brumley and Tuveri remote timing attack [16]. In most cases, the multiplication is done as $(k + q)G$.

Algorithm 2 OpenSSL implementation of kG using wNAF

Require: Scalar k in the wNAF form e_0, e_1, …, e_{λ_l} and precomputed points $\{\pm G, \pm 3G, \ldots, \pm(2^w - 1)G\}$;
Ensure: $Q = kG$;

```
Q ⇐ 0;
for i = λ_l, λ_l − 1, ..., 0 do
    Q ⇐ 2 · Q;
    if e_i ≠ 0 then
        Q ⇐ Q + e_i G;
    end if
end for
```

We can see that in Algorithm 2, when the **if-then** block is ran into, digit e_i is non-zero, and vice versa. So if whether the **if-then** block is executed or cannot be detected during each loop of **for-do**, we can determine whether e_i is zero or not. The Flush+Reload attack [10, 19], which is a new kind

of cache side-channel attack [11] that targets the last-level cache (LLC) shared between different cores, can be used to detect whether the program has ran into the **if-then** block of Algorithm 2 by using a spy program to monitor LLC hits/misses. Denote "A" for an add operation in the **if-then** block, and "D" for a double operation. As a double operation is done every time before the **if-then** block is executed, there is a "D" right before each "A", but we omit these "D"s. Under the assumption of a perfect side-channel, we may obtain a "double-and-add" chain of information as below (assume that the window size $w = 3$, and the sequence is written from the higher digit to the lower digit):

$$\text{“ADDDADDDDDDAD}\ldots\ldots\text{DDDADD”}.$$

In fact, the position of the i-th "A" (counted from the lower index) is just the position of the non-zero digit k_i, which is denoted before as λ_i for $1 \leqslant i \leqslant l$. Note that $\lambda_l + 1$ is the length of the double-and-add chain.

OpenSSL uses the modified wNAF representation in the actual implementation instead of the generalized one as stated in Algorithm 1 to avoid length expansion in some cases and to make exponentiation more efficient. The representation of modified wNAF is very similar to the wNAF. Each non-zero coefficient is followed by at least w zero coefficients, except for the most significant digit which is allowed to violate this condition in some cases. As the use of modified wNAF affects the attack results little, we only consider the case of the wNAF for simplification. In fact, we propose how to deal with the modified wNAF in Subsection 3.1.

2.3 The hidden number problem

Given a prime number q and a positive integer z. Let $t_1, \ldots, t_d$ be randomly chosen, which are uniformly and independently in $\mathbb{F}_q$. For $1 \leqslant i \leqslant d$, let u_i be the value such that $|v_i| = |u_i - \alpha t_i|_q \leqslant q/2^{z+1}$, where $0 < \alpha < q$ is unknown and $|\cdot|_q$ denotes the reduction modulo q into range $[-q/2, \ldots, q/2)$. The HNP is to find the hidden number α.

This problem can be translated to a CVP instance which can be further converted to a shortest vector problem (SVP). This is done by constructing a $(d+2)$-dimensional lattice L spanned by the following matrix:

$$M = \begin{pmatrix} q & & & & & \\ & \ddots & & & & \\ & & q & & & \\ t_1 & \ldots & t_d & 1/2^{z+1} & & 0 \\ u_1 & \ldots & u_d & 0 & & q/2^{z+1} \end{pmatrix}.$$

In the lattice $L = L(M)$, there exists a vector

$$\boldsymbol{w} = (h_1, \ldots, h_d, \alpha, -1) \cdot M = \left(v_1, \ldots, v_d, \alpha/2^{z+1}, -q/2^{z+1}\right) \in L,$$

where h_i is the integer satisfying that $h_i q + \alpha t_i - u_i = v_i$ for $1 \leqslant i \leqslant d$. Its Euclidean norm satisfies that $\|\boldsymbol{w}\| \leqslant \sqrt{d+2} \cdot (q/2^{z+1})$, while the determinant of $L(M)$ is $q^{d+1}/2^{2z+2}$. The ratio of $\|\boldsymbol{w}\|$ and $|L|^{1/(d+2)}$ is thus no greater than $\sqrt{d+2}(q/2^{d(z+1)})^{1/(d+2)}$, which indicates that if z is not too small and d is properly large, $\boldsymbol{w}$ is a very short vector. The second vector in a reduced basis is expected to be equal to $\boldsymbol{w}$ with a "good" chance for a suitably strong lattice reduction algorithm. Note that the first reduced basis vector is likely to be $(-t_1, \ldots, -t_d, q, 0) \cdot M = (0, \ldots, 0, q/2^{z+1}, 0)$.

3 Attacking ECDSA

In this section, we introduce how to utilize only a fraction of the double-and-add chain of the ephemeral key k to extract information and use the obtained information to recover the ECDSA secret key. We first

propose a new way of extracting information that takes advantage of the length of the double-and-add chain and the positions of two non-zero digits to recover information about k. After that, we give an efficient way of utilizing the obtained information, translating the problem of recovering ECDSA secret key to the HNP, which is further converted to the problem of solving approximate SVP.

In the rest of this paper, assume that we have gotten a fraction of the double-and-add chain of the ephemeral key k, say the positions of the second "A" and the last "A" (from the higher index) together with the length of the double-and-add chain. Suppose there are l non-zero digits in the wNAF representation of k. Let k_i be i-th non-zero digit and $\lambda_i \geqslant 0$ be the position of k_i for $1 \leqslant i \leqslant l$. Then we have $k = \sum_{i=1}^{l} k_i 2^{\lambda_i}$. Let L_{AD} be the length of the double-and-add chain of k, and L_0 be the length of its binary string. In the OpenSSL implementation of ECDSA, L_0 is always set to be $\lfloor \log_2 q \rfloor + 1$ as stated in Subsection 2.2.

3.1 Extracting information

In this subsection, we first show how to extract information about k from the MSDs. After that, a combined method using information of both MSDs and LSBs is proposed to extract more information. By writing k as an algebraic expression of a new unknown variable with a smaller range, we are able to extract some information about k. Estimation about the amount of extracted information will be further stated in Subsection 3.3.

As can be observed that there exist three different cases of the double-and-add chain in total considering the length, i.e., $L_{\mathrm{AD}} < L_0$, $L_{\mathrm{AD}} > L_0$ and $L_{\mathrm{AD}} = L_0$. According to Algorithm 2, if $L_{\mathrm{AD}} < L_0$ there must be $L_0 - L_{\mathrm{AD}} < w$, where w is the window size. If $L_{\mathrm{AD}} > L_0$, then there must be $L_{\mathrm{AD}} - L_0 = 1$ (this will be further proved in Proposition 1). So when $w = 3$, all the possible cases are $L_{\mathrm{AD}} - L_0 = 1$, $L_{\mathrm{AD}} = L_0$, $L_0 - L_{\mathrm{AD}} = 1$ and $L_0 - L_{\mathrm{AD}} = 2$. Moreover, it can be easily checked that $\lambda_l = L_{\mathrm{AD}} - 1$ and that $2^{L_0 - 1} \leqslant k \leqslant 2^{L_0} - 1$.

3.1.1 *Utilizing the MSD information*

We first introduce some lemmas and propositions before presenting our results of extracting information about the ephemeral key k utilizing the MSDs. The following two lemmas follow directly from the results of [29].

Lemma 1 ([29]). For integer m satisfying $1 \leqslant m \leqslant l - 1$, we have

$$\left| \sum_{i=1}^{m} k_i \cdot 2^{\lambda_i} \right| < 2^{w + \lambda_m} \leqslant 2^{\lambda_{m+1} - 1}. \tag{2}$$

Lemma 2 ([29]). For all the possible cases of L_{AD}, we have

(1) If $L_{\mathrm{AD}} > L_0$, then $k_{l-1} < 0$ and $k_l = 1$;

(2) If $L_{\mathrm{AD}} = L_0$, then $k_{l-1} > 0$ and $k_l = 1$;

(3) If $L_{\mathrm{AD}} < L_0$, then $2^{L_0 - L_{\mathrm{AD}}} < k_l < 2^{L_0 - L_{\mathrm{AD}} + 1}$.

The proofs of Lemmas 1 and 2 can be found in [29].

From Lemma 1, we can easily get $k_l > 0$. Since if $k_l < 0$, $k < k_l \cdot 2^{\lambda_l} + 2^{\lambda_l - 1} = (1 + 2k_l)2^{\lambda_l - 1} < 0$, which is obviously a contradiction.

The conclusion in Lemma 2 implies that when $w = 3$, if $L_0 - L_{\mathrm{AD}} = 1$, $k_l = 3$; if $L_0 - L_{\mathrm{AD}} = 2$, the only possible values of MSD are $k_l = 5, 7$.

We only consider the wNAF representation here. While in the modified wNAF representation, if $k_l = 1$, $k_{l-1} < 0$, and $\lambda_l - \lambda_{l-1} = w + 1$, the position of k_l is modified from λ_l to $\lambda_l - 1$ to make $L_{\mathrm{AD}} = L_0$. If this case happens, the distance between k_l and k_{l-1} is w, which can be easily detected. So if we find that $L_{\mathrm{AD}} = L_0$, but $\lambda_l - \lambda_{l-1} = w$, then $k_l = 1$ and $k_{l-1} < 0$.

Proposition 1. If $L_{\mathrm{AD}} > L_0$, then there must be $L_{\mathrm{AD}} - L_0 = 1$.

Proof. As k can be rewritten as $k = \sum_{i=1}^{l} k_i \cdot 2^{\lambda_i} = k_l \cdot 2^{\lambda_l} + \sum_{i=1}^{l-1} k_i \cdot 2^{\lambda_i} \geqslant 2^{\lambda_l} + \sum_{i=1}^{l-1} k_i \cdot 2^{\lambda_i}$. If $L_{\mathrm{AD}} - L_0 > 1$, $\lambda_l = L_{\mathrm{AD}} - 1 \geqslant L_0 + 1$. Applying Lemma 1, we have $k \geqslant 2^{\lambda_l} - 2^{\lambda_l - 1} = 2^{\lambda_l - 1} \geqslant 2^{L_0}$, which contradicts with the fact that $k \leqslant 2^{L_0} - 1$.

Next, we will show how to utilize the information of MSDs to extract information about k. We propose the following theorems applying Lemma 2.

Theorem 1. If $L_{\mathrm{AD}} > L_0$, $|k - 2^{\lambda_l} + 2^{w-1} \cdot 2^{\lambda_{l-1}}| < 2^{\lambda_{l-1}+w-1}$.

Proof. From the conclusion (1) in Lemma 2, we have that $k_l = 1$ and $k_{l-1} < 0$. Let $k'_{l-1} = -k_{l-1}$, then $1 \leqslant k'_{l-1} \leqslant 2^w - 1$. So we have

$$k = \sum_{i=1}^{l} k_i \cdot 2^{\lambda_i} = 2^{\lambda_l} - k'_{l-1} \cdot 2^{\lambda_{l-1}} + \sum_{i=1}^{l-2} k_i \cdot 2^{\lambda_i}.$$

From Lemma 1, we have that $|\sum_{i=1}^{l-2} k_i \cdot 2^{\lambda_i}| < 2^{\lambda_{l-1}-1}$. While on the other hand, $-2^{w-1} + 1 \leqslant k'_{l-1} - 2^{w-1} \leqslant 2^{w-1} - 1$, i.e., $|k'_{l-1} - 2^{w-1}| \leqslant 2^{w-1} - 1$. Thus

$$\begin{aligned}\left|k - 2^{\lambda_l} + 2^{w-1} \cdot 2^{\lambda_{l-1}}\right| &= \left|-(k'_{l-1} - 2^{w-1}) \cdot 2^{\lambda_{l-1}} + \sum_{i=1}^{l-2} k_i \cdot 2^{\lambda_i}\right| \\ &< (2^{w-1} - 1) \cdot 2^{\lambda_{l-1}} + 2^{\lambda_{l-1}-1} \\ &< 2^{w-1} \cdot 2^{\lambda_{l-1}} = 2^{\lambda_{l-1}+w-1}.\end{aligned}$$

Remark 1. From Theorem 1, there exists an unknown variable k' that $k = k' + 2^{\lambda_l} - 2^{w-1} \cdot 2^{\lambda_{l-1}}$ and it satisfies that $|k'| < 2^{\lambda_{l-1}+w-1}$. As can be easily checked that, the range of k' is smaller than that of k. Similarly, we can write k as an algebraic expression of a new unknown variable k' with a smaller range in the remaining two theorems and we will omit the explanations for simplification.

Theorem 2. If $L_{\mathrm{AD}} = L_0$, then $|k - 2^{\lambda_l} - 2^{w-1} \cdot 2^{\lambda_{l-1}}| < 2^{\lambda_{l-1}+w-1}$.

Proof. From the conclusion (2) in Lemma 2, we have that $k_l = 1$ and $k_{l-1} > 0$. Then $1 \leqslant k_{l-1} \leqslant 2^w - 1$. So the ephemeral key k can be rewritten as

$$k = \sum_{i=1}^{l} k_i \cdot 2^{\lambda_i} = 2^{\lambda_l} + k_{l-1} \cdot 2^{\lambda_{l-1}} + \sum_{i=1}^{l-2} k_i \cdot 2^{\lambda_i}.$$

Then similar to the proof above, we have

$$\begin{aligned}\left|k - 2^{\lambda_l} - 2^{w-1} \cdot 2^{\lambda_{l-1}}\right| &= \left|(k_{l-1} - 2^{w-1}) \cdot 2^{\lambda_{l-1}} + \sum_{i=1}^{l-2} k_i \cdot 2^{\lambda_i}\right| \\ &< (2^{w-1} - 1) \cdot 2^{\lambda_{l-1}} + 2^{\lambda_{l-1}-1} \\ &< 2^{w-1} \cdot 2^{\lambda_{l-1}} = 2^{\lambda_{l-1}+w-1}.\end{aligned}$$

Theorem 3. Suppose $w = 3$. If $L_0 - L_{\mathrm{AD}} = 1$, then $|k - 3 \cdot 2^{\lambda_l}| < 2^{\lambda_l - 1}$; if $L_0 - L_{\mathrm{AD}} = 2$, $|k - 6 \cdot 2^{\lambda_l}| < 2^{\lambda_l + 1}$.

Proof. From the conclusion (3) in Lemma 2, we have $k_l = 3$ if $L_0 - L_{\mathrm{AD}} = 1$. So k can be rewritten as

$$k = \sum_{i=1}^{l} k_i \cdot 2^{\lambda_i} = 3 \cdot 2^{\lambda_l} + \sum_{i=1}^{l-1} k_i \cdot 2^{\lambda_i}.$$

From Lemma 1, we have

$$\left|k - 3 \cdot 2^{\lambda_l}\right| = \left|\sum_{i=1}^{l-1} k_i \cdot 2^{\lambda_i}\right| < 2^{\lambda_{l-1}+w} \leqslant 2^{\lambda_l - 1}.$$

Similarly, if $L_0 - L_{\mathrm{AD}} = 2$, we have that $k_l = 5$ or 7. So $|k_l - 6| \leqslant 1$. From Lemma 1, we have

$$\left|k - 6 \cdot 2^{\lambda_l}\right| = \left|(k_l - 6) \cdot 2^{\lambda_l} + \sum_{i=1}^{l-1} k_i \cdot 2^{\lambda_i}\right| < 2^{\lambda_l} + 2^{\lambda_l - 1} < 2^{\lambda_l + 1}.$$

3.1.2 *Combining the leaked MSDs and LSBs*

Since the elements in $\{k_i\}_{i=1}^{l}$ are all odd, we have $k_1 = 1 + 2k_1'$, where k_1' is an integer. The ephemeral key can be rewritten as $k = \sum_{i=2}^{l} k_i \cdot 2^{\lambda_i} + (1 + 2k_1') \cdot 2^{\lambda_1} = \sum_{i=2}^{l} k_i \cdot 2^{\lambda_i} + k_1' \cdot 2^{\lambda_1+1} + 2^{\lambda_1}$. We can easily have that $2^{\lambda_1+1} | (k - 2^{\lambda_1})$. What's more, it can be easily seen that if there is an integer A that $2^{\lambda_1+1} | A$, then $2^{\lambda_1+1} | (k \pm A - 2^{\lambda_1})$. The following theorem indicates the way of utilizing information about both the MSDs and the LSBs, which can be used to extract information about k.

Theorem 4. Let A and B be two integers.

(1) If $L_{\mathrm{AD}} > L_0$, $A = 2^{\lambda_1} + 2^{\lambda_l} - 2^{w-1} \cdot 2^{\lambda_{l-1}}$, $B = q/(2^{\lambda_{l-1}+w-\lambda_1-2})$.

(2) If $L_{\mathrm{AD}} = L_0$, $A = 2^{\lambda_1} + 2^{\lambda_l} + 2^{w-1} \cdot 2^{\lambda_{l-1}}$, $B = q/(2^{\lambda_{l-1}+w-\lambda_1-2})$.

(3) Suppose $w = 3$. If $L_0 - L_{\mathrm{AD}} = 1$, $A = 2^{\lambda_1} + 3 \cdot 2^{\lambda_l}$, $B = q/(2^{\lambda_l-\lambda_1-2})$.

(4) Suppose $w = 3$. If $L_0 - L_{\mathrm{AD}} = 2$, $A = 2^{\lambda_1} + 6 \cdot 2^{\lambda_l}$, $B = q/(2^{\lambda_l-\lambda_1})$.

Then we have

$$\left| \frac{k - A}{2^{\lambda_1+1}} \right| \leqslant \frac{q}{B}. \tag{3}$$

Proof. We will prove the inequation (3) in different cases respectively.

(1) Suppose $L_{\mathrm{AD}} > L_0$. From Theorem 1, we have $|k - 2^{\lambda_l} + 2^{w-1} \cdot 2^{\lambda_{l-1}}| < 2^{\lambda_{l-1}+w-1}$. So

$$\left| k - 2^{\lambda_l} - 2^{\lambda_1} + 2^{w-1} \cdot 2^{\lambda_{l-1}} \right| < 2^{\lambda_{l-1}+w-1} + 2^{\lambda_1}.$$

If $l \geqslant 2$, we can easily get that $2^{\lambda_1+1} | (-2^{\lambda_l} + 2^{w-1} \cdot 2^{\lambda_{l-1}})$, so $2^{\lambda_1+1} | (k - 2^{\lambda_l} - 2^{\lambda_1} + 2^{w-1} \cdot 2^{\lambda_{l-1}})$, which means that

$$\left| \frac{k - 2^{\lambda_l} - 2^{\lambda_1} + 2^{w-1} \cdot 2^{\lambda_{l-1}}}{2^{\lambda_1+1}} \right| < 2^{\lambda_{l-1}+w-\lambda_1-2} + \frac{1}{2}.$$

As the ratio on the left of the inequation is always an integer, so we further have

$$\left| \frac{k - 2^{\lambda_l} - 2^{\lambda_1} + 2^{w-1} \cdot 2^{\lambda_{l-1}}}{2^{\lambda_1+1}} \right| \leqslant 2^{\lambda_{l-1}+w-\lambda_1-2}.$$

This finishes the proof of the first case.

(2) Suppose $L_{\mathrm{AD}} = L_0$. From Theorem 2, we have $|k - 2^{\lambda_l} - 2^{w-1} \cdot 2^{\lambda_{l-1}}| < 2^{\lambda_{l-1}+w-1}$. So

$$\left| k - 2^{\lambda_l} - 2^{\lambda_1} - 2^{w-1} \cdot 2^{\lambda_{l-1}} \right| < 2^{\lambda_{l-1}+w-1} + 2^{\lambda_1}.$$

It is easy to check that if $l \geqslant 2$, we have $2^{\lambda_1+1} | (k - 2^{\lambda_l} - 2^{\lambda_1} - 2^{w-1} \cdot 2^{\lambda_{l-1}})$. Similar to the proof above,

$$\left| \frac{k - 2^{\lambda_l} - 2^{\lambda_1} - 2^{w-1} \cdot 2^{\lambda_{l-1}}}{2^{\lambda_1+1}} \right| \leqslant 2^{\lambda_{l-1}+w-\lambda_1-2}.$$

This finishes the proof of the second case.

(3) Suppose $w = 3$ and $L_0 - L_{\mathrm{AD}} = 1$. According to Theorem 3, we have $|k - 3 \cdot 2^{\lambda_l}| < 2^{\lambda_l-1}$. So

$$\left| k - 2^{\lambda_1} - 3 \cdot 2^{\lambda_l} \right| < 2^{\lambda_l-1} + 2^{\lambda_1}.$$

We can easily see that if $l \geqslant 2$, $2^{\lambda_1+1} | (k - 2^{\lambda_1} - 3 \cdot 2^{\lambda_l})$. Similarly, we have

$$\left| \frac{k - 2^{\lambda_1} - 3 \cdot 2^{\lambda_l}}{2^{\lambda_1+1}} \right| \leqslant 2^{\lambda_l-\lambda_1-2}.$$

This finishes the proof of the third case.

(4) Suppose $w = 3$ and $L_0 - L_{\mathrm{AD}} = 2$. According to Theorem 3, we have $|k - 6 \cdot 2^{\lambda_l}| < 2^{\lambda_l+1}$. So $|k - 2^{\lambda_1} - 6 \cdot 2^{\lambda_l}| < 2^{\lambda_l+1} + 2^{\lambda_1}$.

Since we have $2^{\lambda_1+1} | (k - 2^{\lambda_1} - 6 \cdot 2^{\lambda_l})$ if $l \geqslant 2$, similarly, there is

$$\left| \frac{k - 2^{\lambda_1} - 6 \cdot 2^{\lambda_l}}{2^{\lambda_1+1}} \right| \leqslant 2^{\lambda_l-\lambda_1}.$$

This finishes the proof of the last case.

Remark 2. From Theorem 4, there exists an unknown variable $\overline{k}$ that $k = \overline{k} \cdot 2^{\lambda_1+1} + A$ and it satisfies that $|\overline{k}| < q/B$. As can be easily checked that, the range of $\overline{k}$ is smaller than that of k. Roughly speaking, the smaller the range of $\overline{k}$ is, the more information we will extract by our attack.

3.2 Lattice attack

In this subsection, we translate the problem of recovering secret key to solving the HNP using the extracted information about MSDs and LSBs, which is further converted to a SVP problem.

Assume that we capture d signatures (r_i, s_i) of message m_i for $1 \leqslant i \leqslant d$ that share a common private key with the i-th ephemeral key denoted as ek_i. Suppose we get $\lambda_{i,l-1}$, $\lambda_{i,1}$ together with the length of the double-and-add chain of ek_i, where $\lambda_{i,l-1}$ and $\lambda_{i,1}$ are separately the positions of the second and last non-zero digit of the wNAF form of ek_i counted from the higher index. Then we have $ek_i = 2^{\lambda_{i,1}+1} \cdot \overline{k_i} + A_i$ with $|\overline{k_i}| \leqslant q/B_i$, where A_i and B_i are the integers satisfying the conditions in Theorem 4. Note that from (1), we have that $ek_i \equiv \alpha r_i s_i^{-1} + H(m_i)s_i^{-1} \bmod q$. Thus we have the following d congruences:

$$\begin{cases} \overline{k_1} = \alpha r_1 s_1^{-1} 2^{-(\lambda_{1,1}+1)} + (H(m_1)s_1^{-1} - A_1)2^{-(\lambda_{1,1}+1)} + h_1 q, \\ \qquad\qquad\qquad\qquad \cdots\cdots \\ \overline{k_i} = \alpha r_i s_i^{-1} 2^{-(\lambda_{i,1}+1)} + (H(m_i)s_i^{-1} - A_i)2^{-(\lambda_{i,1}+1)} + h_i q, \\ \qquad\qquad\qquad\qquad \cdots\cdots \\ \overline{k_d} = \alpha r_d s_d^{-1} 2^{-(\lambda_{d,1}+1)} + (H(m_d)s_d^{-1} - A_d)2^{-(\lambda_{d,1}+1)} + h_d q, \end{cases} \tag{4}$$

where the values of α $(0 < \alpha < q)$, $\overline{k_i}$ $(|\overline{k_i}| \leqslant q/B_i)$ and h_i are unknown. Using the method in [24, 25], we compute the following values:

$$t_i = \lfloor r_i s_i^{-1} 2^{-(\lambda_{i,1}+1)} \rfloor_q, \quad u_i = \lfloor -(H(m_i)s_i^{-1} - A_i)2^{-(\lambda_{i,1}+1)} \rfloor_q,$$

where $\lfloor \cdot \rfloor_q$ denotes the reduction modulo q into the range $[0, \ldots, q)$. Then we have

$$|\overline{k_i}| = |\alpha t_i - u_i|_q \leqslant \frac{q}{B_i}.$$

Thus we can use (4) to construct a lattice L spanned by the following matrix:

$$M = \begin{pmatrix} B_1 \cdot q & & & & & \\ & \ddots & & & & \\ & & B_i \cdot q & & & \\ & & & \ddots & & \\ & & & & B_d \cdot q & \\ B_1 \cdot t_1 & \cdots & B_i \cdot t_i & \cdots & B_d \cdot t_d & 1 \\ B_1 \cdot u_1 & \cdots & B_i \cdot u_i & \cdots & B_d \cdot u_d & 0 \quad q \end{pmatrix}.$$

In the lattice $L = L(M)$, there exists a vector

$$\boldsymbol{w} = (h_1, \ldots, h_i, \ldots, h_d, \alpha, -1) \cdot M = \left(B_1 \cdot \overline{k_1}, \ldots, B_i \cdot \overline{k_i}, \ldots, B_d \cdot \overline{k_d}, \alpha, -q\right) \in L.$$

Its Euclidean norm satisfies that $\|\boldsymbol{w}\| \leqslant \sqrt{d+2} \cdot q$, while the determinant of $L(M)$ is $\prod_{i=1}^{d} B_i \cdot q^{d+1}$. As explained in Subsection 2.3, the second vector in a reduced basis of M is expected to be equal to $\boldsymbol{w}$ if the value of $(\prod_{i=1}^{d} B_i)^{1/d}$ is not too small and d is properly large, while the first reduced basis vector is likely to be $(-t_1, \ldots, -t_d, q, 0) \cdot M = (0, \ldots, 0, q, 0)$.

3.3 Attack analysis

One important factor that may influence the attack result is how much information we can get from the extracting method. As we can see from the attack in Subsection 3.2, the size of $\overline{k}$ (corresponding to ephemeral key k) is of vital importance to the attack result. The smaller the range value of $\overline{k}$ is relative to q, the better the attack result will be. Denote the range value of $\overline{k}$ as K, then we call $\log_2(q/K) - 1$ as the number of leaked bits. In order to estimate the number of leaked bits, we give the following theorem.

Theorem 5. Let N_l be the expected number of leaked bits from ephemeral key k via our attack.

(1) If $L_{\mathrm{AD}} > L_0$, then $N_l = \log_2 q - L_0 + 4 \geqslant 3$.

(2) If $L_{\mathrm{AD}} = L_0$, then $N_l = \log_2 q - L_0 + 5 \geqslant 4$.

(3) Suppose $w = 3$. If $L_0 - L_{\mathrm{AD}} = 1$, then $N_l = \log_2 q - L_0 + 4 \geqslant 3$.

(4) Suppose $w = 3$. If $L_0 - L_{\mathrm{AD}} = 2$, then $N_l = \log_2 q - L_0 + 3 \geqslant 2$.

Proof. Let us first calculate the expected values of λ_1 and $\lambda_l - \lambda_{l-1}$. The probability for λ_1 being 0 is 1/2, for 1 is 1/4, for 2 is 1/8, ... Thus the expected value of λ_1 is $\sum_{i=1}^{\infty}(i-1)/2^i \approx 1$. Similarly for $\lambda_l - \lambda_{l-1}$, as $\lambda_l - \lambda_{l-1} \geqslant w+1$, the probability for $\lambda_l - \lambda_{l-1}$ being $w+1$ is 1/2, for $w+2$ is 1/4, for $w+3$ is 1/8, ... Thus the expected value of $\lambda_l - \lambda_{l-1}$ is $w+1+\sum_{i=1}^{\infty}(i-1)/2^i \approx w+2$.

As $L_0 = \lfloor \log_2 q \rfloor + 1$, we can easily get that $L_0 - 1 \leqslant \log_2 q < L_0$.

(1) If $L_{\mathrm{AD}} > L_0$, we have $L_0 = L_{\mathrm{AD}} - 1$ according to Proposition 1, thus $L_0 = \lambda_l$. From Theorem 4,

$$\left|\overline{k}\right| \leqslant \frac{q}{B} = 2^{\lambda_{l-1}+w-\lambda_1-2} = K.$$

Since $\log_2(q/K) - 1 = \log_2 q - \log_2 2^{\lambda_{l-1}+w-\lambda_1-2} - 1 = (\log_2 q - \lambda_l) + (\lambda_l - \lambda_{l-1}) - w + \lambda_1 + 1$, so the expected number of leaked bits $N_l = \log_2 q - \lambda_l + 4 = \log_2 q - L_0 + 4$. As $\log_2 q \geqslant L_0 - 1$, we further have $N_l \geqslant 3$.

(2) When $L_{\mathrm{AD}} = L_0$, we have $L_0 = \lambda_l + 1$. Thus

$$\left|\overline{k}\right| \leqslant \frac{q}{B} = 2^{\lambda_{l-1}+w-\lambda_1-2} = K.$$

Since $\log_2(q/K) - 1 = \log_2 q - \log_2 2^{\lambda_{l-1}+w-\lambda_1-2} - 1$, so $N_l = \log_2 q - \lambda_l + 4 = \log_2 q - L_0 + 5$. Similarly we further have that $N_l \geqslant 4$.

(3) Suppose $w = 3$ and $L_0 - L_{\mathrm{AD}} = 1$. We have $L_0 = L_{\mathrm{AD}} + 1 = \lambda_l + 2$. According to Theorem 4, there is

$$\left|\overline{k}\right| \leqslant \frac{q}{B} = 2^{\lambda_l-\lambda_1-2} = K.$$

Since we have that $\log_2(q/K) - 1 = \log_2 q - \log_2 2^{\lambda_l-\lambda_1-2} - 1$. Thus $N_l = \log_2 q - L_0 + 4$, which further gives that $N_l \geqslant 3$.

(4) Suppose $w = 3$ and $L_0 - L_{\mathrm{AD}} = 2$. We have $L_0 = L_{\mathrm{AD}} + 2 = \lambda_l + 3$. So

$$\left|\overline{k}\right| \leqslant \frac{q}{B} = 2^{\lambda_l-\lambda_1} = K.$$

Since we have that $\log_2(q/K) - 1 = \log_2 q - \log_2 2^{\lambda_l-\lambda_1} - 1$. Thus $N_l = \log_2 q - L_0 + 3$. This further gives that $N_l \geqslant 2$.

Remark 3. If q is quite close to 2^{L_0}, say $q \approx 2^{L_0} - \epsilon$ where $\epsilon > 0$ is small, then $\log_2 q - L_0$ is quite close to 0, thus the value of N_l for the above four cases will separately be about 4, 5, 4 and 3, i.e., 1 bit more than the general cases. And this is the case when ECDSA is implemented on curve **secp256k1**.

4 Experiment results

We mount our attack to the **secp256k1** curve with window size being $w = 3$. According to our statistical results, the probability of ephemeral keys satisfying the four cases in Theorem 4 for a 256-bit ECDSA signature is separately 40.175%, 19.7%, 19.45% and 20.675%. So from Remark 3, the average number of leaked bits we can get via our new method is about $4\times40.175\%+5\times19.7\%+4\times19.45\%+3\times20.675\% = 3.99$ per signature. In theory, it requires 65 signatures to recover the secret key using a (65+2)-dimensional lattice since $65 \times 3.99 > 256$. However, due to the limited ability of lattice reduction algorithm and other implementation reasons, the actual number of required signatures may be more than 65. When we use about 100 signatures to attack ECDSA, the lattice dimension turns out to be a little time-consuming for the BKZ algorithm to find a short vector, meanwhile the success probability is not that high. So we seek for some improvements by making some selection on signatures.

4.1 Signature selection

Basically, the average number of leaked bits of per signature together with the lattice dimension influence the success probability of our attack. Roughly, the more bits per signature leaks, the lower the lattice dimension will be, thus it will be easier for a lattice reduction algorithm to recover the secret key. To increase the average number of leaked bits, one straight idea is to select those signatures who can leak more information. By selecting signatures carefully, we are able to recover a ECDSA secret key with a smaller lattice dimension, which leads directly to a save in the consuming time. What's more, we can get a higher success probability with the same lattice dimension.

While on the other hand, the total number of required signatures is the ratio of the number of selected signatures and the probability of a signature satisfying the selection rule. If we increase the average number of leaked bits by selecting signatures, the total number of signatures may be more. So we need to make a compromise between the average number of leaked bits and the total number of signatures. The following introduces three selection methods which are independent to each other and can be used in combination.

4.1.1 *Constraints on length of the double-and-add chain*

As the signature that satisfies $L_0 - L_{\mathrm{AD}} = 2$ when $w = 3$ leaks 3 bits on average, which seems to be less than the other cases. So we discard these signatures. The average number of leaked bits will be raised to about $(4 \times 40.175\% + 5 \times 19.7\% + 4 \times 19.45\%)/(1 - 20.675\%) \approx 4.248$, while the probability of a signature satisfying our selection rule being $1 - 20.675\% = 79.325\%$. So in theory, we can recover the secret key using a lattice with dimension being $256/4.248 + 2 \approx 63$ and the average number of signatures we need in total is about $61/79.325\% \approx 77$. As stated before, the actual number of required signatures might be more than 77. The following two cases are similar.

4.1.2 *Constraints on MSDs*

We constrain the value of $\lambda_l - \lambda_{l-1}$ to be not less than $w + 2$ to get more leaked bits when $L_{\mathrm{AD}} \geqslant L_0$. The average number of leaked bits for the first two cases of Theorem 4 will separately become about 5 and 6. We can get that the probability that a signature meets our requirement will be about $(40.175\% + 19.7\%) \times 50\% + 19.45\% + 20.675\% \approx 70.07\%$, and the average number of leaked bits will becomes about 4.27. We can use a (60+2)-dimensional lattice to recover the secret key in theory while the average number of signatures we need in total is about 86.

Suppose we set $\lambda_l - \lambda_{l-1} \geqslant w + 3$ whose probability is 25%. Similarly we can get that the probability that a signature meets our requirement will be about 55.10% and the average number of leaked bits is about $(6 \times 40.175\% \times 25\% + 7 \times 19.7\% \times 25\% + 4 \times 19.45\% + 3 \times 20.675\%)/55.10\% \approx 4.26$. So in theory, the lattice required in our attack is with dimension being $256/4.26 + 2 \approx 62$, and the average number of signatures we need in total is about $60/55.10\% \approx 109$. Notice that when the constraint of value of $\lambda_l - \lambda_{l-1}$ is set to $w + 3$ instead of $w + 2$, the expected number of leaked bits is actually not increased. So we do not consider this case.

4.1.3 *Constraints on LSBs*

We can also control the number of zeros in the LSBs as in [22]. Denote the length of the known run of zeros in the LSBs of k as z. A constraint is set to the value of z such that $z \geqslant Z$ where $Z \geqslant 0$ is a determined integer, i.e., if there are less than Z consecutive zeros in the LSBs of k, the signature would be discarded. In fact, if we set $Z = 1$, the LSBs are expected to leak about $1 + \sum_i^{\infty} i/2^i \approx 3$ bits of information on average. Thus the average number of leaked bits is raised by 1 theoretically, being about 4.99 bits. And the probability of signatures satisfying this condition is 50%. So in theory, we can recover the secret key using a lattice with dimension being $52 + 2 = 54$ and the average number of signatures in total is about 104.

Table 1 Comparisons of the selection rules

Selection rule	Average number of leaked bits	Lattice dimensions (in theory)	Signature number (in theory)	Signature Pr.[a)] (%)
0	3.990	67	65	100
1	4.248	63	77	79.325
2	4.800	56	110	49.400
3	4.990	54	104	50.000

a) Signature Pr. denotes the probability of a signature satisfying the selection rule.

Our choice. Applying the above three selection methods, we may increase the number of leaked bits, lower the lattice dimensions and improve the attack results with a sacrifice of a little more signatures.

In order to balance the lattice dimension with the total number of signatures, we propose the following selection rules which can be used independently in our experiments:

Rule 0. All signatures are used without any discard.

Rule 1. Throw away those signatures that satisfy $L_0 - L_{\mathrm{AD}} = 2$ when $w = 3$.

Rule 2. First throw away those signatures satisfying $L_0 - L_{\mathrm{AD}} = 2$ when $w = 3$. Then if $L_{\mathrm{AD}} \geqslant L_0$, select those satisfying $\lambda_l - \lambda_{l-1} \geqslant w + 2$; if $L_0 - L_{\mathrm{AD}} = 1$, signatures are preserved without selection.

Rule 3. Select those signatures that $z \geqslant 1$ where z is the length of the known run of zeros in the LSBs of k.

Rule 1 is just the selection method of constraints on L_{AD}; Rule 2 is the combination of the selection methods of constraints on L_{AD} together with constraints on MSDs; Rule 3 is the selection method of constraints on LSBs. Comparisons of the four rules are stated in Table 1. The average number of leaked bits, the lattice dimensions, the number of signatures and the probability of a signature satisfying the selection rule can all be computed as above.

4.2 Attack results

Our attack is mounted to curve **secp256k1** on an Intel Core i7-3770 CPU running at 3.40 GHz in single thread. It does not rely on any special property of curves, thus can be applied freely to any other curve that use the wNAF algorithm to implement scalar multiplications. The BKZ algorithm implemented in the NTL library written in C++ is used to implement the lattice reduction. In our experiments, we use the BKZ algorithm with blocksize 25 and 30. The results would be surely improved with a greater blocksize or by using a more effective lattice reduction algorithm, like BKZ2.0 [27]. Given the selection rule and the number of selected signatures (which is value of the lattice dimension minus 2), we execute 200 experiments each time to test the success probability. Some of the results are listed in Table 2. The experiment results show that, given the lattice dimension selection rule 1, 2 and 3 can get a higher success probability compared to rule 0.

Compared to the best result of [22] that succeeds recovering the ECDSA secret key using 200 signatures with probability being 3.5% (implemented by BKZ-30), our best result manages to succeed with only 85 signatures with probability being 1.5% and 90 signatures with probability being 5%, which is much better. Note that our attack can be applied to different side-channel attacks. In [30], for example, totally 1278 signatures are needed to recover the ECDSA secret key (mounted to curve **secp256k1**), utilizing only the information of LSBs of the ephemeral key. But if our method is applied, the number of signatures can be cut down to nearly 85.

5 Conclusion

In this paper, we mount an attack on ECDSA implemented by the latest version of OpenSSL which uses the wNAF method to execute the scalar multiplication. Our attack is based on just a small proportion of the double-and-add chain, which can be obtained by the Flush+Reload attack but not limited to it. We

Table 2 Experiment results[a)]

Lattice dimension	Selection rule	Number of signatures	Reduction algorithm	p (%)	Average time (min)
67	0	65	BKZ-30	×	×
	1	82		×	×
	2	132		16	13.67
	3	130		16.5	
72	0	70	BKZ-30	×	×
	1	89		×	×
	2	142		35	20.96
	3	140		30.5	
77	0	75	BKZ-30	×	×
	1	94		0.5	35.78
	2	152		36.5	
	3	150		42	
82	0	80	BKZ-30	×	×
	1	101	BKZ-25	1.5	7.58
	2	166		42	
	3	160		46.5	
87	0	85	BKZ-30	1.5	26.97
	1	107	BKZ-25	6	10.48
	2	172		49.5	
	3	170		52.5	
92	0	90	BKZ-25	5	15.77
	1	113		25	
	2	182		44.5	
	3	180		50	

a) "p" denotes the success probability of recovering the secret key; "Average time" denotes the average time that the algorithm cost each time; "Selection rule" denotes the choice of selecting signatures as Subsection 4.1.3 states, and Rule "0" means that no selection is made on signatures.

develop a new extraction method to retrieve information about the ephemeral key from the double-and-add chain, utilizing only the positions of two non-zero digits and the length of the chain. By combining the information of both the most significant digits and the LSBs, we are able to obtain no less than 2.99 bits of information per signature on average, and even about 3.99 bits on average for $q \approx 2^n - \epsilon$ where $\epsilon > 0$ is small. Mounted to the **secp256k1** curve, our attack can successfully recover a ECDSA secret key using only 85 signatures.

Acknowledgements This work was supported by National Basic Research Program of China (973 Program) (Grant No. 2013CB338003).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Bellare M, Canetti R, Krawczyk H. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, 1998. 419–428
2 Blake-Wilson S, Menezes A. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In: Proceedings of the 5th International Workshop on Security Protocols, Paris, 1998. 137–158
3 Diffie W, van Oorschot P C, Wiener M J. Authentication and authenticated key exchanges. Design Code Cryptoger, 1992, 2: 107–125
4 National Institute of Standards and Technology. Digital signature standard (DSS). FIPS PUB 186. http://csrc.nist.gov/publications/PubsFIPS.html
5 National Institute of Standards and Technology. Digital signature standard (DSS). FIPS PUB 186-4. http://csrc.nist.

gov/publications/fips/fips186-3
6 Johnson D, Menezes A, Vanstone S A. The elliptic curve digital signature algorithm (ECDSA). Int J Inf Secur, 2001, 1: 36–63
7 Vanstone S. Responses to NIST's proposal. Commun ACM, 1992, 35: 50–52
8 Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. 2008. http://www.cryptovest.co.uk/resources/Bitcoin%20paper%20Original.pdf
9 The openssl project. OpenSSL — cryptography and SSL/TLS toolkit. Version 1.0.2h. 2016
10 Yarom Y, Benger N. Recovering OpenSSL ECDSA nonces using the Flush+Reload cache side-channel attack. IACR Cryptology ePrint Archive, 2014, 140. http://eprint.iacr.org/
11 Kocher P C, Jaff J, Jun B. Differential power analysis. In: Proceedings of the 19th Annual International Cryptology Conference, Santa Barbara, 1999. 388–397
12 Page D. Theoretical use of cache memory as a cryptanalytic side-channel. IACR Cryptology ePrint Archive 2002, 2002: 169. http://eprint.iacr.org/
13 Acıiçmez O, Koç Ç K, Seifert J P. On the power of simple branch prediction analysis. In: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, Singapore, 2007. 312–320
14 Brumley B B, Hakala R M. Cache-timing template attacks. In: Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, 2009. 667–684
15 Tromer E, Osvik D A, Shamir A. Efficient cache attacks on AES, and countermeasures. J Cryptol, 2010, 23: 37–71
16 Brumley B B, Tuveri N. Remote timing attacks are still practical. In: Proceedings of the 16th European Symposium on Research in Computer Security, Leuven, 2011. 355–371
17 Zhang Y, Juels A, Reiter M K, et al. Cross-VM side channels and their use to extract private keys. In: Proceedings of the ACM Conference on Computer and Communications Security, Raleigh, 2012. 305–316
18 Irazoqui G, Inci M S, Eisenbarth T, et al. Lucky 13 strikes back. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, Singapore, 2015. 85–96
19 Yarom Y, Falkner K. Flush+Reload: a high resolution, low noise, L3 cache side-channel attack. In: Proceedings of the 23rd USENIX Security Symposium (USENIX Security 2014), San Diego, 2014. 719–732
20 Allan T, Brumley B B, Falkner K, et al. Amplifying side channels through performance degradation. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, Los Angeles, 2016. 422–435
21 Hlaváč M, Rosa T. Extended hidden number problem and its cryptanalytic applications. In: Proceedings of the 13th International Conference on Selected Areas in Cryptography, Montreal, 2006. 114–133
22 Benger N, van de Pol J, Smart N P, et al. "Ooh aah... just a little bit": a small amount of side channel can go a long way. In: Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded System, Busan, 2014. 75–92
23 Howgrave-Graham N, Smart N P. Lattice attacks on digital signature schemes. Design Code Cryptoger, 2001, 23: 283–290
24 Nguyen P Q, Shparlinski I. The insecurity of the digital signature algorithm with partially known nonces. J Cryptol, 2002, 15: 151–176
25 Nguyen P Q, Shparlinski I. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. Design Code Cryptoger, 2003, 30: 201–217
26 Liu M, Nguyen P Q. Solving BDD by enumeration: an update. In: Proceedings of Cryptographers' Track at the RSA Conference, San Francisco, 2013. 293–309
27 Chen Y, Nguyen P. BKZ2.0: better lattice security estimates. In: Proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, 2011. 1–20
28 van de Pol J, Smart N P, Yarom Y. Just a little bit more. In: Proceedings of Cryptographer's Track at the RSA Conference, San Francisco, 2015. 3–21
29 Fan S, Wang W, Cheng Q. Attacking OpenSSL implementation of ECDSA with a few signatures. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, 2016. 1505–1515
30 Genkin D, Pachmanov L, Pipman I, et al. ECDSA key extraction from mobile devices via nonintrusive physical side channels. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, 2016. 1626–1638