Efficient Comb Elliptic Curve Multiplication Methods Resistant to Power Analysis

Min Feng[1], Bin B. Zhu[2], Maozhi Xu[1], Shipeng Li[2]

[1]School of Mathematical Sciences, Peking Univ., Beijing,100871, China

{fengmin, mzxu}@math.pku.edu.cn

[2]Microsoft Research Asia, Beijing, 100080, China

{binzhu, spli}@microsoft.com

Corresponding Author:

Bin B. Zhu

Microsoft Research Asia,

3F Sigma, No. 49, Zhichun Road, Haidian,

Beijing, 100080, China

Phone: (8610) 62617711 ext. 3109

Fax: (8610) 88097306

Email: binzhu@ieee.org (preferred)

or binzhu@microsoft.com

Efficient Comb Elliptic Curve Multiplication Methods Resistant to Power Analysis

Min Feng, Bin B. Zhu, Maozhi Xu, Shipeng Li

Abstract

Elliptic Curve Cryptography (ECC) has found wide applications in smart cards and embedded systems. Point multiplication plays a critical role in ECC. Many efficient point multiplication methods have been proposed. One of them is the comb method [5] which is much more efficient than other methods if precomputation points are calculated in advance or elsewhere. Unfortunately, Many efficient point multiplication methods including the comb method are vulnerable to power-analysis attacks. Various algorithms to make elliptic curve point multiplication secure to power-analysis attacks have been proposed recently, such as the double-and-add-always method [8], Möller's window method [17, 18], Okeya et al.'s odd-only window method [21, 22], and Hedabou et al.'s comb method [19]. In this paper, we first present a novel comb recoding algorithm which converts an integer to a sequence of signed, odd-only comb bit-columns. Using this recoding algorithm, we then present several comb methods, both Simple Power Analysis (SPA)-nonresistant and SPA-resistant, for point multiplication. These comb methods are more efficient than the original SPA-nonresistant comb method and Hedabou et al.'s SPA-resistant comb method. Our comb methods inherit the advantage of a comb method, running much faster than Möller's window method and Okeya et al.'s odd-only window method, as well as other window methods such as the efficient signed m-ary window method, if only the evaluation phase is taken into account. Combined with randomization projective coordinates or other randomization techniques and certain precautions in selecting elliptic curves and parameters, our

SPA-resistant comb methods are resistant to all power-analysis attacks.

Keywords: comb method, power-analysis attack, side-channel attack, SPA-resistant, elliptic curve cryptography, ECC, power analysis, SPA, DPA, scalar multiplication, point multiplication, signed odd-only comb method, smart cards.

1 Introduction

Elliptic curve Cryptography (ECC) has gained increasing popularity in public key cryptography since it was first proposed by Miller [1] and Koblitz [2]. ECC exploits the fact that there is no sub-exponential algorithm to solve the discrete logarithm problem on elliptic curves. Compared with other public key cryptography such as RSA [3], ECC utilizes shorter key sizes for the same level of security, which translates into fast computation and less demands on memory and CPU. These advantages make ECC ideal for use in smart cards and embedded systems where storage, power, and computing resources are at a premium.

A major component in ECC is point (or scalar) multiplication. Many efficient point multiplication methods have been developed: binary method, non-adjacent form (NAF) method, and several window methods that play tradeoffs between storage space and execution time. A good description of most point multiplication methods can be found in the book [4]. A method that is not included in the book is the comb method first proposed by Lim and Lee [5], which uses a binary matrix to represent a scalar and processes the matrix columnwise. The comb method can dramatically speed up the main computation of point multiplication with the same precomputation space as window methods.

Unfortunately, all the aforementioned methods are vulnerable to side-channel attacks, first introduced by Kocher et al. [6, 7], which measure observable parameters such as timings or power consumptions during cryptographic operations to deduce the whole or partial secret information of a cryptosystem. Power analysis includes both Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [7]. In SPA, a single trace of power consumption for execution of cryptographic operations is inspected to extract information about the cryptographic operations and the secret key. DPA is much more powerful and sophisticated. In DPA, statistical analysis and error correction techniques are used to analyze many traces of power consumptions to extract information correlated to the secret key. Side-channel attacks were extended to elliptic curve cryptosystems [8]. Higher order DPA attacks are proposed in [9, 10]. Goubin [11] proposed a refined DPA attack by using a special point on the elliptic curve with one of coordinates being zero. A particular target of side-channel attacks for elliptic curve cryptosystems is the scalar k in point multiplication which computes a product kP, where P is a point on an elliptic curve $E(F)$ over a finite field F and k is a secret positive integer. With power analysis, partial information or the exact value of the secret k can be deduced when Lim and Lee's comb method [5] or the point multiplication methods described in the book [4] are used.

Many countermeasures have been proposed to protect against side-channel attacks on ECC. Two major strategies are used to thwart SPA attacks. The first is to make addition and doubling operations indistinguishable. A unified formula for computing both addition and doubling has been proposed by Liardet and Smart [12] for Jacobi-type and by Joye and Quisquater [13] for Hasse-type elliptic curves. A common disadvantage of these methods is that the group order of the elliptic curve has to possess certain properties, which renders them inapplicable to the elliptic curves recommended by the National Institute of Standards and Technology (NIST) [14].

The second strategy is to remove dependency on a specific value of the secret multiplier k in the intermediate steps of point multiplication. Coron, Okeya, et al. [8, 15, 23, 25, 24] proposed schemes using addition chains to always execute point addition and doubling for each bit. Möller, Okeya, et al. [17, 18, 21, 22] modified window methods by making addition chains with a fixed pattern of nonzero digits. Hedabou et al. [19] proposed a first SPA-resistant comb method. Chevallier-Mames et al. [26] proposed a scheme which, through inserted dummy operations, divides the two basic operations in point multiplication, point doubling and point addition, into side-channel atomic blocks so that point multiplication appears as a succession of side-channel atomic blocks that are indistinguishable by SPA.

An SPA-resistant method is not necessarily resistant to DPA attacks, but many countermeasures have been proposed to transform an SPA-resistant method into a DPA-resistant method. Coron [8] proposed to use random projective coordinates. Joye and Tymen [20] proposed to use a random isomorphism such as a random elliptic curve isomorphism and a random field isomorphism.

In this paper, we present a novel comb recoding algorithm to convert a scalar into a signed, odd-only fixed pattern comb representation. Using this recoding algorithm, we then present a new comb method called *Signed Odd-Only Comb Method* and its variations which compute point multiplication more efficiently with less precomputed points than the original comb method proposed in [5]. Those comb methods are then modified to be SPA-resistant by exploiting the fact that point addition and point subtraction are virtually of the same computational complexity for elliptic curves. Our SPA-resistant comb methods are more efficient in both storage space and execution time than the SPA-resistant comb method proposed by Hedabou et al. [19]. They also inherit a comb method's advantage – running much faster than SPA-resistant window methods when precomputation points are calculated in advance or elsewhere. Combined with the techniques proposed elsewhere that convert an SPA-resistant method into a DPA-resistant method, our SPA-resistant comb methods to be presented in this paper are secure to all power analysis attacks. The advantages offered by our comb methods are very desirable for smart cards and embedded systems where power and computing resources are at a premium.

This paper is organized as follows. In the next section, we introduce preliminaries for elliptic curve cryptosystems and efficient elliptic curve point multiplication methods. In Section 3, side-channel attacks and existing SPA-resistant point multiplication methods related to the methods to be presented in this paper are described. Our novel comb recoding algorithms and comb methods are presented in Section 4. Security analysis is also provided in this section, along with performance comparison with other point multiplication methods. The paper is concluded in Section 6.

2 Preliminaries

2.1 Elliptic Curves Equations

An elliptic curve over a field F can be expressed by its Weierstrass form:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad a_i \in F.$$

We denote by $E(F)$ the set of points $(x, y) \in F^2$ satisfying the above equation plus the "point at infinity" $\mathcal{O}$. With the chord-tangent process [4], $E(F)$ forms an ablelian group with the point at infinity $\mathcal{O}$ as the zero, and point addition as the group's binary operation. Given two points P_1 and P_2 in $E(F)$, a third point $P_3 = P_1 + P_2 \in E(F)$ as the addition of P_1 and P_2 can be calculated. A special point addition that a point adds itself is called *doubling*. The cost of point doubling is usually different from that of point addition. Point addition and doubling need to compute costly field inversions. By using the Jacobian projective coordinates which represent a point $P = (x, y)$ as $P = (X, Y, Z)$, where $x = X/Z^2$ and $y = Y/Z^3$, and the infinity point $\mathcal{O}$ as $(\theta^2, \theta^3, 0)$, $\theta \in F^*$, field inversions can be avoided at the expense of more field multiplications. A field multiplication is usually much faster than a field inversion, resulting in faster elliptic curve point addition and doubling.

The group $E(F)$ generated by an elliptic curve over some finite field F meets the public key cryptography requirements that the discrete logarithm problem is very difficult to solve. Therefore

ECC has been used in many standards and applications. Elliptic curves used in cryptography are elliptic curves defined over fields F_{2^m} or fields F_p where m is a large number and p is a big prime. Over these two types of fields, the Weierstrass form reduces to the short Weierstrass form, and point addition and doubling are also simplified. In practical applications, elliptic curves are usually required to contain a subgroup of a large prime order ρ. An elliptic curve cryptosystem typically employs only points in this ρ-order subgroup of $E(F)$. For details of elliptic curve equations and point operations, interested readers are referred to [4].

2.2 Point Multiplication

Adding a point P to itself k times is called *point multiplication* or *scalar multiplication*, and is denoted as $Q = kP$, where k is a positive integer. Many efficient algorithms have been proposed for point multiplication, as described in the book [4]. One of them is the m-ary method which represents an integer k with the fixed-pattern m-ary representation, where m is usually equal to 2^w for some integer $w \geqslant 2$. The m-ary method splits point multiplication into precomputation stage and evaluation stage to trade storage space for execution speed.

Algorithm 1. *m-ary Method.*

```
Input: A point $P$, an integer $k = \sum_{i=0}^{d-1} a_i m^i, a_i \in \{0, 1, \cdots, m-1\}$
Output: $Q = kP$.
Precomputation Stage:
1. $P_1 = P$.
2. For $i = 2$ to $m-1$ do: $P_i = P_{i-1} + P$.
3. $Q = \mathcal{O}$.
Evaluation Stage:
4. For $i = d-1$ to $0$ by $-1$ do:
5.     $Q = mQ$, which requires $w$ doublings,
6.     $Q = Q + P_{a_i}$.
7. Return $Q$.
```

The precomputation stage in the m-ary method needs to store $m - 1 \equiv 2^w - 1$ points. In our storage estimation in this paper, the input point P is always included. Let us estimate the time cost for the method. In our cost estimation, operations involving $\mathcal{O}$ are not counted. The precomputation stage needs to calculate $2P, 3P, \cdots, [2^w - 1]P$. Since the total number of doublings and additions in calculating those points remains the same, and doubling is more efficient than addition with projective coordinates which are typically used in applications, we would like to use doubling operations as many as possible in this stage. The most efficient scheme is to calculate in the following way:

$$\begin{cases} P_{2i} &= 2P_i, \\ \\ P_{2i+1} &= P_{2i} + P. \end{cases} \tag{1}$$

Using this scheme, the precomputation costs $(2^{w-1} - 1)D + (2^{w-1} - 1)A$, where D means point doubling and A means point addition. As for the time cost for the evaluation stage, if we assume the most significant digit is not zero, i.e., $a_{d-1} \neq 0$, then the number of doublings in this stage is $w(d-1)$. If $a_i = 0, i \neq d-1$, then the addition in Step 6 is not needed. Assuming that k is uniformly distributed, the average number of additions in the evaluation stage is $\frac{2^w-1}{2^w}(d-1)$. Therefore the average time cost in the evaluation stage is approximately $\{w(d-1)\}D + \{\frac{2^w-1}{2^w}(d-1)\}A$, and the average total time cost including both stages is approximately $\{2^{w-1} - 1 + w(d-1)\}D + \{2^{w-1} - 1 + \frac{2^w-1}{2^w}(d-1)\}A$. The m-ary method can be modified to be more efficient. They can also be extended to the signed m-ary window method. Details of these methods can be found in [4].

In 1994, Lim and Lee proposed a comb method [5] which can also calculate point multiplication efficiently. Let $k = \sum_{i=0}^{n-1} b_i 2^i$ be an n-bit integer, where $b_i \in \{0, 1\}$. For an integer $w \geqslant 2$, set $d = \lceil \frac{n}{w} \rceil$. We define

$$[b_{w-1}, b_{w-2}, \cdots, b_1, b_0] \equiv b_{w-1}2^{(w-1)d} + b_{w-2}2^{(w-2)d} + \cdots + b_1 2^d + b_0,$$

where $(b_{w-1}, b_{w-2}, \cdots, b_1, b_0) \in {Z_2}^w$. The comb method uses a binary matrix of w rows and d columns to represent an integer k, and processes the matrix columnwise.

Algorithm 2. *Fixed-base Comb Method*

```
**Input:** A point $P$, an integer $k = \sum_{i=0}^{n-1} b_i 2^i$, $b_i \in \{0,1\}$, and a window width $w \geqslant 2$.
**Output:** $Q = kP$.
Precomputation Stage:
1. Compute $[b_{w-1}, \cdots, b_1, b_0]P$ for all $(b_{w-1}, \cdots, b_1, b_0) \in Z_2^w$.
2. Write $k = K^{w-1}||\cdots||K^1||K^0$, padding with 0 on the left if necessary,
where each $K^j$ is a bit-string of length $d$. Let $K_i^j$ denote the $i$-th bit of $K^j$.
Define $\mathbb{K}_i \equiv [K_i^{w-1}, \cdots, K_i^1, K_i^0]$.
3. $Q = \mathcal{O}$.
Evaluation Stage:
4. For $i = d-1$ to $0$ by $-1$ do:
5.      $Q = 2Q$,
6.      $Q = Q + \mathbb{K}_i P$.
7. Return $Q$.
```

The comb method stores $2^w - 1$ points in the precomputation stage. Let us estimate its time cost. In the precomputation stage, $[b_{w-1}, \cdots, b_1, b_0]P$ needs to be calculated for $(b_{w-1}, \cdots, b_1, b_0) \in Z_2^w$. An efficient scheme is given in [19]: First $2^d P, 2^{2d} P, \cdots, 2^{(w-1)d} P$ are calculated. The cost is $(w-1)d$ doubling operations. Next all possible combinations with only two nonzero bits in $[b_{w-1}, \cdots, b_1, b_0]P$ are calculated from the results of doubling operations. There are C_w^2 such combinations. Each combination uses one point addition. There are C_w^2 point additions in this step. In the next step, all combinations with three nonzero bits are calculated. There are C_w^3 such combinations. Each needs one point addition from the previously calculated results. Therefore this step costs C_w^3 point additions. This procedure continues until all the precomputation points have been calculated. The total number of point additions in the precomputation stage is therefore

$$\sum_{i=2}^{w} C_w^i = \sum_{i=0}^{w} C_w^i - C_w^1 - C_w^0 = 2^w - w - 1.$$

In conclusion, the time cost in the precomputation stage is

$$\{(w-1)d\}D + \{(2^w - w - 1)\}A.$$

To estimate the time cost in the evaluation stage, we assume that the most significant bit-column of $\{\mathbb{K}_i\}$ is not zero, i.e., $\mathbb{K}_{d-1} \neq 0$. Then the number of doubling operations in the evaluation stage is $(d-1)$. If $\mathbb{K}_i = 0, i \neq d-1$, then the point addition in Step 6 is not needed. If we assume

k is uniformly distributed, the probability that $\mathbb{K}_i \neq 0$ is $\frac{2^w-1}{2^w}$, and the average number of point additions is $\frac{2^w-1}{2^w}(d-1)$. Therefore the average time cost in the evaluation stage is approximately $\{(d-1)\}D+\{\frac{2^w-1}{2^w}(d-1)\}A$. This cost is much smaller than the time cost in the evaluation stage for the m-ary method, which is $\{w(d-1)\}D+\{\frac{2^w-1}{2^w}(d-1)\}A$ as described previously. This gain at the evaluation stage is at the cost of higher time cost at the precomputation stage for the comb method. The total time cost including both stages is $\{(w-1)d+(d-1)\}D+\{(2^w-w-1)+\frac{2^w-1}{2^w}(d-1)\}A = \{wd-1\}D+\{(2^w-w-1)+\frac{2^w-1}{2^w}(d-1)\}A$ for the comb method.

3 Side-Channel Attacks and Countermeasures

3.1 Side-Channel Attacks

Two types of power analysis have been introduced by Kocher [6, 7]. One is Simple Power Analysis (SPA). The other is Differential Power Analysis (DPA).

Simple Power Analysis. SPA analyzes a single trace of power consumption in a crypto-device during point multiplication. A branch instruction condition can be identified from the recorded power consumption data. This represents continuity of elliptic curve doubling operation. If the double-and-add method [4] is used in computing point multiplication, each bit of the secret multiplier k is revealed by this attack. For other point multiplication methods such as the m-ary and comb methods, though SPA cannot deduce the value of a digit a_i for the m-ary method or $\mathbb{K}_i$ for the comb method, it can detect if a digit a_i or $\mathbb{K}_i$ is zero or not, which means leak of secret information.

Differential Power Analysis. DPA records many power traces of point multiplications, and uses correlation among the records and an error correction technique [7] to deduce some or all digits of the secret k. DPA is more complex yet more powerful than SPA. An SPA-resistant point multiplication method is not necessarily resistant to DPA attacks, but many countermeasures can be used to transform an SPA-resistant method to a DPA-resistant method. A common practice is to make execution, and thus power consumption, different for identical inputs. Randomization is usually employed to achieve this effect. All the following randomizing approaches are feasible: randomizing an input point in projective coordinates, randomizing an exponential parameter representation, randomizing an elliptic curve equation, and randomizing a field representation. This paper focuses on SPA-resistant point multiplication. All these randomizing approaches can be applied to transform our SPA-resistant methods to be also resistant to DPA attacks.

3.2 Related Existing SPA-Resistant Methods

Many countermeasures to SPA attacks have been proposed. A particular approach is to make execution of point multiplication independent of any specific value of the multiplier k. The simplest method is the double-and-add-always method proposed by Coron [8] which removes the branching operation in point multiplication by adding dummy operations so that the same operations are applied no matter the current bit is 0 or 1. Three more efficient SPA-resistant methods following that particular approach are described below. They all divide point multiplication into two stages: precomputation stage and evaluation stage. Our methods also follow that approach.

3.2.1 HPB's Comb Method

Hedabou, Pinel and Bénéteau (HPB) [19] proposed recently a comb method to thwart SPA attacks. The main idea is to extend $\mathbb{K}_i$ in the comb method Alg. 2 to a signed representation $(\mathbb{K}'_i, s_i)$, where each $\mathbb{K}'_i$ is nonzero, and s_i is a sign. The following procedure is used to obtain such a signed representation $(\mathbb{K}'_i, s_i)$ for an odd integer k represented by $\mathbb{K}_i, 0 \leqslant i < d$, in the comb method. Let $s_0 = 1$. The rest is constructed as follows,

$$\begin{cases} \begin{array}{ll} (\mathbb{K}'_i, s_i) & = (\mathbb{K}_{i-1}, s_{i-1}) \\ (\mathbb{K}'_{i-1}, s_{i-1}) & = (\mathbb{K}_{i-1}, -s_{i-1}) \end{array} & \text{if } \mathbb{K}_i = 0 \\ \begin{array}{ll} (\mathbb{K}'_i, s_i) & = (\mathbb{K}_i, s_{i-1}) \\ (\mathbb{K}'_{i-1}, s_{i-1}) & = (\mathbb{K}_{i-1}, s_{i-1}) \end{array} & \text{otherwise.} \end{cases}$$

HPB's comb method applies this signed representation to the conventional comb method Alg. 2 to calculate $(k+1)P$ for even k and $(k+2)P$ for odd k. $2P$ is then calculated. P or $2P$ is subtracted from the result of the conventional comb method to obtain the desired point kP.

HPB's comb method has the same time and space costs in the precomputation stage as the original comb method, i.e., storage of $2^w - 1$ points and time cost of $\{(w-1)d\}D + \{(2^w - w - 1)\}A$. The evaluation stage costs $d-1$ point additions and $d-1$ doublings. The last stage after the conventional comb method costs one doubling and one subtraction. Therefore the total cost of the HPB's method including both precomputation and evaluation stages is $(w-1)d + (d-1) + 1 = wd$ doubling operations and $(2^w - w - 1) + (d-1) + 1 = 2^w - w + d - 1$ adding operations. Compared with the conventional comb method Alg. 2, HPB's method has the same storage cost and a little higher time cost.

3.2.2 Möller's m-ary Method

Möller proposed an SPA-resistant m-ary point multiplication method [18]. For an integer $k = \sum_{i=0}^{d} a_i 2^{wi}$ with $a_i \in \{0, 1, \cdots, 2^w - 1\}$, Möller's method first converts k to another representation $k = \sum_{i=0}^{d'} a'_i 2^{wi}$ such that $a'_i \in \{-2^w, \pm 1, \pm 2, \cdots, \pm(2^{w-1} - 1), 2^{w-1}\}$ where d' is either d or $d+1$. Intuitively, this recoding algorithm replaces 0 digits by -2^w and adjusts the next more significant digit to keep k unchanged. The recoding algorithm is expressed recursively with two auxiliary values c_i and t_i, $0 \leqslant c_i \leqslant 2$ and $0 \leqslant t_i \leqslant 2^w + 1$. Set $c_0 = 0$. For $i = 0, \cdots, d+1$, let

$$t_i = a_i + c_i$$

and

$$(c_{i+1}, a'_i) = \begin{cases} (1, -2^w) & \text{if } t_i = 0 \\ (0, t_i) & \text{if } 0 < t_i < 2^{w-1} \\ (1, -2^w + t_i) & \text{if } 2^{w-1} < t_i < 2^w \\ (2, -2^w) & \text{if } t_i = 2^w \\ (1, 1) & \text{if } t_i = 2^w + 1. \end{cases}$$

Note that the equation $c_{i+1} \cdot 2^w + a'_i = t_i$ always holds.

After the conversion, Möller's point multiplication method is exactly the same as the m-ary method Alg. 1 for $m = 2^w$ except that the precomputation stage needs to calculate $2P, 3P, \cdots, [2^{w-1}]P$, and $[-2^w]P$. These points are calculated with Eq. 1, i.e., using the same scheme as in the m-ary method. The cost of the precomputation stage is then $(2^{w-2}+1)D + (2^{w-2}-1)A$. The evaluation stage costs $w(d-1)D + (d-1)A$ if $d' = d$ or $wdD + dA$ if $d' = d+1$. Therefore the total cost of Möller's method including both stages is at least

$$(2^{w-2} + wd - w + 1)D + (2^{w-2} + d - 2)A.$$

The precomputation stage in Möller's method stores $2^{w-1} + 1$ points.

3.2.3 Odd-Only m-ary Method

Okeya and Takagi (OT) proposed an SPA-resistant odd-only m-ary point multiplication method [21]. The main idea is to convert an odd integer $k = \sum_{i=0}^{d} a_i 2^{wi}$ with $a_i \in \{0, 1, \cdots, 2^w - 1\}$ to another representation $k = \sum_{i=0}^{d} a'_i 2^{wi}$ such that $a'_i \in \{\pm 1, \pm 3, \cdots, \pm(2^w - 1)\}$. This can be achieved with the following recoding algorithm developed independently by us yet equivalent to OT's recoding algorithm.

Algorithm 3. *SPA-Resistant Odd-Only m-ary Recoding Algorithm for an Odd Scalar.*

```
Input: An odd n-bit integer $k = \sum_{i=0}^{d} a_i 2^{wi} > 0$ with $a_i \in \{0, 1, \cdots, 2^w - 1\}$.
Output: $k = \sum_{i=0}^{d} a'_i 2^{wi}$ with $a'_i \in \{\pm 1, \pm 3, \cdots, \pm(2^w - 1)\}$.
1. for $i = 0$ to $d - 1$ by 1 do:
2.     if $a_i$ is odd, then set $a'_i = a_i$,
3.     if $a_i$ is even, then set $a'_i = a_i + 1$ and $a'_{i-1} = a'_{i-1} - 2^w$.
```

Using this conversion, the m-ary method Alg. 1 is used with $m = 2^w$ to calculate $[k + 1]P$ for even k and $[k+2]P$ for odd k. P or $2P$ is then subtracted from the result to obtain the desired point kP. OT's method needs to store 2^{w-1} points $P, 3P, 5P, \cdots, [2^w - 1]P$. These points are calculated by computing first $2P$ and then the rest iteratively with the equation $[i]P = 2P + [i-2]P$. The cost in this stage is $1D + (2^{w-1} - 1)A$. The *for*-loop in the evaluation stage costs $w(d-1)D + (d-1)A$, and the post-*for*-loop processing costs one doubling to calculate $2P$ and one subtraction to subtract either P or $2P$. The total cost of OT's method including both stages is therefore

$$(wd - w + 2)D + (2^{w-1} + d - 1)A.$$

4 Our Odd-Only Comb Method

4.1 Recoding Algorithm

Like HPB's comb method, our approach is also to transform all the comb bit-columns $\{\mathbb{K}_i\}$ of a scalar k into a signed nonzero representation $\{\mathbb{K}'_i \neq 0\}$. The major difference between our method and HPB's method is that every $\mathbb{K}'_i$ generated by our novel recoding method is a signed odd number. More specifically, our recoding scheme generates $K_i'^0 \in \{1, \bar{1}\}$ and $K_i'^j \in \{0, K_i'^0\}$, $j \neq 0$ for each bit-column $\mathbb{K}'_i \equiv [K_i'^{w-1}, \cdots, K_i'^1, K_i'^0]$, where $\bar{1}$ is defined as -1. The advantage of our recoding method over other comb recoding methods is that the precomputation stage needs to compute and store only half of the points of other comb methods. The detail of our recoding algorithm is described next for a window width $w \geqslant 2$ and $d = \lceil \frac{n+1}{w} \rceil$.

Algorithm 4. *Signed Odd-Only Comb Recoding Algorithm for an Odd Scalar.*

```
**Input:** An odd n-bit integer $k = \sum_{i=0}^{n-1} b_i 2^i$ with $b_i \in \{0,1\}$.
**Output:** $k = \sum_{i=0}^{wd-1} b'_i 2^i \equiv K'^{w-1}||\cdots||K'^1||K'^0$, padding with 0 on the left if necessary,
where each $K'^j$ is a binary string of $d$ bits long. Let $K'^j_r$ denote the $r$-th bit of $K'^j$, i.e., $K'^j_r \equiv b'_{jd+r}$.
Define $\mathbb{K}'_r \equiv [K'^{w-1}_r, \cdots, K'^1_r, K'^0_r]$. The output satisfies $K'^0_r \in \{1, \bar{1}\}$ and $K'^j_r \in \{0, K'^0_r\}$ for $j \neq 0$
and $0 \leqslant r < d$.
1. for $i = 0$ to $d-1$ by 1 do:
2.     if $b_i = 1$ then set $b'_i = 1$,
3.     if $b_i = 0$ then set $b'_i = 1$ and $b'_{i-1} = \bar{1}$.
4. set $e = \lfloor \frac{k}{2^d} \rfloor$ and $i = d$
5. while $i < wd$ do
6.     if $e$ is odd and $b'_{i \bmod d} = \bar{1}$, then set $b'_i = \bar{1}$ and $e = \lfloor \frac{e}{2} \rfloor + 1$
7.     else set $b'_i = e \mod 2$, and $e = \lfloor \frac{e}{2} \rfloor$
8.     $i = i + 1$
```

The recoding algorithm first converts each one of the last d bits to either 1 or $\bar{1}$ by exploiting the fact that $1 \equiv 1\bar{1}\bar{1}\cdots\bar{1}$, and the least significant bit is 1 for an odd k. In other words, the least significant bit K'^0_r in each bit-column K'_r, $0 \leqslant r < d$, is either 1 or $\bar{1}$. The rest of the recoding algorithm processes each bit at a time from the lowest bit towards the highest bit, starting from the d-th bit. If the current bit is 1 and has a different sign from that of the least significant bit in the same $\mathbb{K}'_r$, the current bit is set to $\bar{1}$ and the value consisting of the remaining higher bits is added by 1 to keep the value of k unchanged. This process generates wd bits $\{b'_i\}$ to represent an odd n-bit integer k.

Theorem 1. *Given an odd scalar k, Algorithm 4 outputs a bit string $\{b'_i\}$ and a sequence of bit-columns $\{\mathbb{K}'_r \equiv [K'^{w-1}_r, \cdots, K'^1_r, K'^0_r]\}$ such that $k = \sum_{i=0}^{wd-1} b'_i 2^i$, $K'^0_r \in \{1, \bar{1}\}$, and $K'^j_r \in \{0, K'^0_r\}$, $j \neq 0$ for each bit-column $\mathbb{K}'_r$ where $0 \leqslant r < d$ and $K'^j_r \equiv b'_{jd+r}$.*

Proof. It is easy to check that each $\mathbb{K}'_r$ generated by Alg. 4 satisfies the conditions that $K'^0_r \in \{1, \bar{1}\}$ and $K'^j_r \in \{0, K'^0_r\}$ for $j \neq 0$: Since k is odd, $b_0 = 1$. Steps 1–3 in Alg. 4 set the least significant bit b'_r in each $\mathbb{K}'_r$, $0 \leqslant r < d$, to be either 1 or $\bar{1}$. Therefore $K'^0_r \equiv b'_r \in \{1, \bar{1}\}, 0 \leqslant r < d$. If $b'_{i \bmod d} = \bar{1}$, b'_i is set to either $\bar{1}$ in Step 6 or 0 in Step 7. If $b'_{i \bmod d} = 1$, b'_i is set to either 0 or 1 in Step 7. This means that all the bits except the least significant bit in each $\mathbb{K}'_r$ are either 0 or of the same value as the the least significant bit in the same $\mathbb{K}'_r$.

To prove $k = \sum_{i=0}^{wd-1} b'_i 2^i$, we first prove

$$\sum_{i=0}^{d-1} b_i 2^i = \sum_{i=0}^{d-1} b'_i 2^i. \tag{2}$$

This can be done by induction. The equation $\sum_{i=0}^{j} b_i 2^i = \sum_{i=0}^{j} b'_i 2^i$ holds for $j = 0$ since k is odd. If the equation is true for $j < d - 1$, then Steps 2 and 3 in Alg. 4 ensure that the equation is also true for $j + 1$. By setting $j = d - 1$, we have the desired equation.

Denote the value of e as e_i when it comes into the i-th loop before Step 6, where $i \geqslant d$. We assert that

$$e_i 2^i + \sum_{j=0}^{i-1} b'_j 2^j = k \tag{3}$$

is always true for $i \geqslant d$. This can be done by induction. By using Eq. 2, we have $k = e_d 2^d + \sum_{i=0}^{d-1} b_i 2^i = e_d 2^d + \sum_{i=0}^{d-1} b'_i 2^i$. This proves that Eq. 3 holds for $i = d$. Assume that Eq. 3 is true for $i \geqslant d$. If e_i is odd and $b'_{i \bmod d} = \bar{1}$, we have $b'_i = \bar{1}$ and $e_{i+1} = \lfloor \frac{e_i}{2} \rfloor + 1 = \frac{e_i - 1}{2} + 1$.

$$e_{i+1} 2^{i+1} + \sum_{j=0}^{i} b'_j 2^j = (\frac{e_i - 1}{2} + 1) 2^{i+1} - 2^i + \sum_{j=0}^{i-1} b'_j 2^j = e_i 2^i + \sum_{j=0}^{i-1} b'_j 2^j = k$$

The same procedure can be used to prove that $e_{i+1} 2^{i+1} + \sum_{j=0}^{i} b'_j 2^j = e_i 2^i + \sum_{j=0}^{i-1} b'_j 2^j = k$ when e_i is even or $b'_{i \bmod d} = 1$. This means that Eq. 3 is also true for $i + 1$. Therefore Eq. 3 is always true for $i \geqslant d$.

The last thing we need to prove is $e_{wd} = 0$. Because k is an integer of n bits, $e_d = \lfloor \frac{k}{2^d} \rfloor$ is an integer of $n - d$ bits: $e_d < 2^{n-d}$. We would like to use induction to prove that for $n \geqslant i \geqslant d$

$$e_i \leqslant 2^{n-i}. \tag{4}$$

We have already proved it when $i = d$. Suppose it is true for a certain i, $n > i \geqslant d$. If e_i is odd, Ineq. 4 implies that $e_i \leqslant 2^{n-i} - 1$. In this case, Steps 6–7 give

$$e_{i+1} \leqslant \lfloor \frac{e_i}{2} \rfloor + 1 \leqslant \lfloor \frac{2^{n-i} - 1}{2} \rfloor + 1 = 2^{n-(i+1)} - 1 + 1 = 2^{n-(i+1)},$$

i.e., Ineq. 4 is true for $i+1$ in this case. If e_i is even, then from Step 7,

$$e_{i+1} \leqslant \lfloor \frac{e_i}{2} \rfloor \leqslant \lfloor \frac{2^{n-i}}{2} \rfloor = 2^{n-(i+1)}.$$

Inqq. 4 still holds. In other words, Ineq. 4 is true for $i+1 \leqslant n$. Therefore Ineq. 4 is proved to be true for $n \geqslant i \geqslant d$. Ineq. 4 derives that $e_n \leqslant 2^0 = 1$.

Since $d = \lceil \frac{n+1}{w} \rceil$, we have $n+1 \leqslant wd$. If $n+1 = wd$, then $e_{wd-1} \equiv e_n \leqslant 1$. If $n+1 < wd$, then from Steps 6–7, $e_{n+1} \leqslant \lfloor \frac{e_n}{2} \rfloor + 1 \leqslant \lfloor \frac{1}{2} \rfloor + 1 = 1$. Continuing with this process, we also have $e_{wd-1} \leqslant 1$. In other words, we always have $e_{wd-1} \leqslant 1$. From Steps 2–3, we have $b'_{d-1} = 1$. When $i = wd-1$ in the loop of Steps 5–8, Step 7 is executed since $b'_{wd-1 \mod d} = b'_{d-1} = 1$, i.e., $e_{wd} = \lfloor \frac{e_{wd-1}}{2} \rfloor \leqslant \lfloor \frac{1}{2} \rfloor = 0$. Applying this result to Eq. 3 yields the desired result $k = \sum_{i=0}^{wd-1} b'_i 2^i$. □

4.2 Signed Odd-Only Comb Methods without Considering SPA

Our recoding method Algorithm 4 works only with odd scalars. If a scalar k is an even number, we first calculate the point multiplication for the odd scalar $k' = k+1$, and then subtract P from the result to obtain the desired result kP.

Algorithm 5. *Signed Odd-Only Comb Method.*

```
Input: A point $P$ and an integer $k$.
Output: $Q = kP$.
Precomputation Stage:
1. Compute $[b_{w-1}, \cdots, b_2, b_1, 1]P$ for all $(b_{w-1}, \cdots, b_2, b_1) \in (Z_2)^{w-1}$.
2. If $k$ is even then $k' = k+1$; otherwise $k' = k$.
3. Apply Alg. 4 to $k'$ to compute the corresponding bit-columns $\mathbb{K}'_0, \mathbb{K}'_1, \cdots, \mathbb{K}'_{d-1}$.
4. $Q = \mathcal{O}$.
Evaluation Stage:
5. for $i = d-1$ to 0 by $-1$ do:
6.     $Q = 2Q$,
7.     $Q = Q + \mathbb{K}'_i P$.
8. if $k$ is even then return $Q - P$ else return $Q$.
```

If the least significant bit of $\mathbb{K}'_i$ is $\bar{1}$, we have $\mathbb{K}'_i = -|\mathbb{K}'_i|$. In this case, Step 7 in Alg. 5 actually executes $Q = Q - |\mathbb{K}'_i|P$.

In practical ECC applications, only elliptic curve points in a subgroup with a large prime order ρ are actually used. In this case, the signed odd-only comb method Alg. 5 can be modified to remove the post-processing Step 8 by exploiting that facts that $\rho - k$ is odd for even k, and $[\rho - k]P = -kP$. This modified method is described as follows.

Algorithm 6. *Signed Odd-Only Comb Method for a Point of Odd Order.*

```
Input: A point $P$ of odd order $\rho$, and an integer $k$.
Output: $Q = kP$.
Precomputation Stage:
1. Compute $[b_{w-1}, \cdots, b_2, b_1, 1]P$ for all $(b_{w-1}, \cdots, b_2, b_1) \in (Z_2)^{w-1}$.
2. If $k$ is odd, set $k' = k$, else set $k' = \rho - k$.
3. Applying Alg. 4 to $k'$ to compute the corresponding bit-columns
$\mathbb{K}'_0, \mathbb{K}'_1, \cdots, \mathbb{K}'_{d-1}$ corresponding to $k'$.
4. $Q = \mathcal{O}$.
Evaluation Stage:
5. for $i = d-1$ to $0$ by $-1$ do:
6.     $Q = 2Q$,
7.     set $Q = Q + (-1)^{k+1}\,\mathbb{K}'_i P$;
8. Return $Q$.
```

Note that in Step 7 above, Q is set to $Q + \mathbb{K}'_i P$ for odd k or $Q - \mathbb{K}'_i P$ for even k.

In our recoding method Alg. 4, d is defined as $\lceil \frac{n+1}{w} \rceil$ instead of $\lceil \frac{n}{w} \rceil$ used in the original comb method Alg. 2. If n is indivisible by w, then d in our recoding method is exactly the same as that in the original comb method, i.e., $\lceil \frac{n+1}{w} \rceil = \lceil \frac{n}{w} \rceil$. But if n is divisible by w, our method's d is one larger than the d used in Alg. 2, i.e., $\lceil \frac{n+1}{w} \rceil = 1 + \lceil \frac{n}{w} \rceil$. Increasing d by one would lead to $w - 1$ additional doublings in the precomputation stage, and one additional addition and one additional doubling in the evaluation stage. Any additional operations are undesirable. Fortunately, most of the additional operations incurred when n is divisible by w can be eliminated by playing a trick as described in the following modified comb method.

Algorithm 7. *Signed Odd-Only Comb Method for n divisible by w.*

```
Input: A point $P$ and an $n$-bit integer $k$.
Output: $Q = kP$.
Precomputation Stage:
1. Compute $[b_{w-1}, \cdots, b_2, b_1, 1]P$ for all $(b_{w-1}, \cdots, b_2, b_1) \in (Z_2)^{w-1}$.
2. If $k \mod 4 = 0$, set $k' = k/2 + 1$;If $k \mod 4 = 1$, set $k' = \lceil k/2 \rceil$;
   If $k \mod 4 = 2$, set $k' = k/2$; If $k \mod 4 = 3$, set $k' = \lfloor k/2 \rfloor$.
3. Applying Alg. 4 to $k'$ to compute the corresponding bit-columns $\mathbb{K}'_0, \mathbb{K}'_1, \cdots, \mathbb{K}'_{d-1}$.
4. $Q = \mathcal{O}$.
Evaluation Stage:
5. for $i = d - 1$ to $0$ by $-1$ do:
6.      $Q = 2Q$,
7.      $Q = Q + \mathbb{K}'_i P$.
8. if $k \mod 4 = 0$, set $Q = Q - P$.
9. $Q = 2Q$.
10. If $k \mod 4 = 1$, set $Q = Q - P$; If $k \mod 4 = 3$, set $Q = Q + P$.
11. Return $Q$.
```

Due to Step 2, the value of d used in Alg. 7 is equivalent to $\lceil \frac{n}{w} \rceil$, the same as the original comb method Alg. 2. Compared with Alg. 5, Alg. 7 saves $w - 1$ doublings in the precomputation stage when n is divisible by w. In this case, one addition is also saved in the evaluation stage if $k \mod 4 = 2$. More performance comparisons of various methods will be given later in this paper.

4.3 SPA-Resistant Signed Odd-Only Comb Method

The comb methods Alg. 5 and Alg. 7 are not SPA-resistant. Even though all the bit-columns $\{\mathbb{K}'_i\}$ are nonzero in both methods, the value of the least significant bit of a scalar k is detectable by SPA in Step 8 of Alg. 5, and information of the last two bits of a scalar k may leak out to SPA in the steps from Step 8 on in Alg. 7. Since all $\mathbb{K}_i \neq 0$, the operations in the *for*-loop are a sequence of alternative point doubling and point addition, $DADA \cdots DADA$, for all the three comb methods described in Section 4.2, therefore do not leak any information about the secret scalar k to SPA. This implies that Alg. 6 is an SPA-resistant comb method if we take every scalar k (or more specifically k' in Step 2 of the algorithm) as an number of $\lceil \log_2 \rho \rceil$ bits, where ρ is the order of the point P. That is a typical assumption in studying SPA-resistant methods. By inserting dummy operations after the *for*-loop, we can easily convert the preceding SPA-nonresistant methods to

SPA-resistant methods. Alg. 5 can be modified to the following SPA-resistant comb method.

Algorithm 8. *SPA-Resistant Signed Odd-Only Comb Method.*

```
Input: A point $P$ and an integer $k$.
Output: $Q = kP$.
Precomputation Stage:
1. Compute $[b_{w-1}, \cdots, b_2, b_1, 1]P$ for all $(b_{w-1}, \cdots, b_2, b_1) \in (Z_2)^{w-1}$.
2. If $k$ is even then $k' = k + 1$, else $k' = k + 2$.
3. Apply Alg. 4 to $k'$ to compute the corresponding bit-columns $\mathbb{K}'_0, \mathbb{K}'_1, \cdots, \mathbb{K}'_{d-1}$.
4. $Q = \mathcal{O}$.
Evaluation Stage:
5. for $i = d - 1$ to 0 by $-1$ do:
6.     $Q = 2Q$,
7.     $Q = Q + \mathbb{K}'_i P$.
8. $P' = 2P$.
9. if $k$ is even then return $Q - P$ else return $Q - P'$.
```

As we have just mentioned, in studying SPA-resistant methods a scalar is considered as a number of $\lceil \log_2 \rho \rceil$ bits where ρ is the order of the point P, i.e., $n = \lceil \log_2 \rho \rceil$ in Alg. 4. In this case, if $\lceil \log_2 \rho \rceil$ is divisible by w, the value of d used in Alg. 8 is one larger than the d in the original comb method Alg. 2, resulting in higher computational complexity. The following SPA-resistant method does not increase d in this case, and therefore removes the increased computational complexity.

Algorithm 9. *SPA-Resistant Signed Odd-Only Comb Method for $n = \lceil \log_2 \rho \rceil$ divisible by w.*

```
Input: A point $P$ of order $\rho$, and an $n$-bit integer $k$.
Output: $Q = kP$.
Precomputation Stage:
1. Compute $[b_{w-1}, \cdots, b_2, b_1, 1]P$ for all $(b_{w-1}, \cdots, b_2, b_1) \in (Z_2)^{w-1}$
2. If $k > \frac{\rho}{2}$ set $k^* = \rho - k$; else set $k^* = k$.
3. If $k^*$ is even then $k' = k^* + 1$, else $k' = k^* + 2$.
4. Apply Alg. 4 to $k'$ to compute the corresponding bit-columns $\mathbb{K}'_0, \mathbb{K}'_1, \cdots, \mathbb{K}'_{d-1}$.
5. $Q = \mathcal{O}$.
Evaluation Stage:
6. for $i = d - 1$ to 0 by $-1$ do:
7.     $Q = 2Q$,
8.     if $k > \frac{\rho}{2}$ then $Q = Q - \mathbb{K}'_i P$ else $Q = Q + \mathbb{K}'_i P$.
9. $P' = 2P$.
10. if $k > \frac{\rho}{2}$ then set $\Delta = 1$ else set $\Delta = -1$.
11. if $k^*$ is even set $Q = Q + \Delta P$ else set $Q = Q + \Delta P'$.
```

Step 2 in Alg. 9 ensures that k^* is less than or equal to $\frac{\rho}{2}$. k' should be a number of $\lceil \log_2 \rho \rceil - 1$ bits to ensure that d used in the method is equal to $\frac{\lceil \log_2 \rho \rceil}{w}$. If not, we can set k' to $k^* - 1$ for even k^* and $k^* - 2$ for odd k^* in Step 3 to achieve the desired result. In this case, Step 10 is modified to set Δ to -1 for $k > \frac{\rho}{2}$ and 1 otherwise.

4.4 Security against Power Analysis

The security of our SPA-resistant comb point multiplication methods is discussed in this section. We first consider the security against SPA, and then describe how to transform our comb methods to resist DPA, the second-order DPA, and other side-channel attacks.

Like other SPA-resistant methods [19, 17, 18, 21, 22], our SPA-resistant comb methods exploit the fact that point subtraction is virtually the same as point addition for power analysis. In addition, these comb methods perform one point addition (or point subtraction) and one doubling at each loop, and the same post-*for*-loop steps disregarding the scalar's value. This means that the same sequence is executed no matter what value a scalar k actually is. Therefore SPA cannot extract any information about the secret k by examining power consumption of the executions. In other words, our SPA-resistant comb methods are really SPA-resistant.

Like other SPA-resistant point multiplication methods, our SPA-resistant comb methods are not necessarily resistant to DPA attacks. Randomization projective coordinates or random isomorphic curves can be used to transform those SPA-resistant methods into DPA-resistant methods.

Okeya and Sakurai's second-order DPA attack [10] might still be successfully applied to our SPA-resistant comb methods when the just-mentioned randomization schemes are used. This second order attack exploits the correlation between power consumption and hamming weight of the loaded data to determine which $\mathbb{K}'_i$ is loaded. To thwart this second-order DPA attack, we can use the same scheme proposed in [19] to protect HPB's comb method – to randomize all precomputed points after getting the point in the table so that there is no fixed hamming weight.

Goubin [11] recently proposed a refined DPA attack on many randomization schemes. This attack employs special points with one of coordinates being zero. To deal with Goubin's DPA attack, a simply way is to choose elliptic curves

$$E : y^2 = x^3 + ax + b$$

defined over F_p $(p > 3)$ with b not being a quadratic residue modulo p, and to reject any point $(x, 0)$ as an input point in applications of our SPA-resistant comb methods. If the cardinality $\#E(F_p)$ is a big prime number, points $(x, 0)$ cannot be eligible input points since they are not on elliptic curves. Combining with the aforementioned randomization techniques and measures, our comb methods can thwart all power-analysis attacks.

4.5 Efficiency

All our comb methods require storage of 2^{w-1} points. In the precomputation stage of our comb methods, $2^d P, 2^{2d} P, \cdots, 2^{(w-1)d} P$ are first calculated. This costs $(w-1)d$ point doublings. Then all possible combinations $[b_{w-1}, b_{w-2}, \cdots, b_2, b_1, 1]P$ with $(b_{w-1}, b_{w-2}, \cdots, b_2, b_1) \in {Z_2}^{w-1}$ are calculated in the same way as the precomputaion stage for Alg. 2, which costs $2^{w-1} - 1$ point additions. The cost of our comb methods in the precomputation stage is therefore $\{(w-1)d\}D + \{2^{w-1} - 1\}A$. The time cost in the evaluation stage varies a little for different comb methods due to the post-*for*-loop processing steps. Assume that the scalar k is randomly distributed, then the average cost in the evaluation stage is $(d-1)D + (d - \frac{1}{2})A$ for Alg. 5, $(d-1)D + (d-1)A$ for Alg. 6, $dD + (d - \frac{1}{4})A$ for Alg. 7, and $dD + dA$ for both Alg. 8 and Alg. 9.

We would like to compare our fixed-base comb methods described in Section 4.2 with the original fixed-base comb method Alg. 2. Table 1 lists the space and time costs for those methods. Our comb methods in Table 1 store 2^{w-1}, which is about half of the stored $2^w - 1$ points in the original comb method Alg. 2. In addition, our comb methods Algs. 5 – 7 save $2^{w-1} - w$ point additions than the original comb method Alg. 2 in the precomputation stage. The evaluation stage has a similar time cost for all the four methods in Table 1. If we want to maintain about the same storage space

Table 1: **Comparison of space and average time costs for the original fixed-base comb method and our fixed-base comb methods described in Section 4.2.**

	Original Comb	Alg. 5	Alg. 6	Alg. 7
d	$\lceil \frac{n}{w} \rceil$	$\lceil \frac{n+1}{w} \rceil$	$\lceil \frac{n+1}{w} \rceil$	$\frac{n}{w}$ where $w \mid n$
Storage	$2^w - 1$	2^{w-1}	2^{w-1}	2^{w-1}
Pre-Stage	$(w-1)dD$ $(2^w - w - 1)A$	$(w-1)dD$ $(2^{w-1} - 1)A$	$(w-1)dD$ $(2^{w-1} - 1)A$	$(w-1)dD$ $(2^{w-1} - 1)A$
Eva-Stage	$(d-1)D$ $\frac{2^w-1}{2^w}(d-1)A$	$(d-1)D$ $(d-\frac{1}{2})A$	$(d-1)D$ $(d-1)A$	dD $(d-\frac{1}{4})A$
Total Cost	$(wd-1)D$ $(2^w - w - 1 + \frac{2^w-1}{2^w}(d-1))A$	$(wd-1)D$ $(2^{w-1} + d - \frac{3}{2})A$	$(wd-1)D$ $(2^{w-1} + d - 2)A$	wdD $(2^{w-1} + d - \frac{5}{4})A$

Table 2: **Comparison of space and average time costs for SPA-resistant comb methods** $(n = \lceil \log_2 \rho \rceil)$
.

	HPB's Comb	Alg. 6	Alg. 8	Alg. 9
d	$\lceil \frac{n}{w} \rceil$	$\lceil \frac{n+1}{w} \rceil$	$\lceil \frac{n+1}{w} \rceil$	$\lceil \frac{n}{w} \rceil$
Storage	$2^w - 1$	2^{w-1}	2^{w-1}	2^{w-1}
Pre-Stage	$(w-1)dD$ $(2^w - w - 1)A$	$(w-1)dD$ $(2^{w-1} - 1)A$	$(w-1)dD$ $(2^{w-1} - 1)A$	$(w-1)dD$ $(2^{w-1} - 1)A$
Eva-Stage	dD dA	$(d-1)D$ $(d-1)A$	dD dA	dD dA
Total Cost	wdD $(2^w - w + d - 1)A$	$(wd-1)D$ $(2^{w-1} + d - 2)A$	wdD $(2^{w-1} + d - 1)A$	wdD $(2^{w-1} + d - 1)A$

for pre-computed points, our methods Algs. 5 – 7 can choose the value of w as $w_2 = w_1 + 1$, one larger than the value $w = w_1$ used in the original comb method, resulting in a similar storage (2^{w_1} v.s. $2^{w_1} - 1$) as the original comb method Alg. 2 yet faster computation in the evaluation stage, thanks to smaller d used in our methods.

Now let us compare our SPA-resistant comb methods with HPB's SPA-resistant comb method. The space and time costs for those SPA-resistant methods are listed in Table 2. Again, our SPA-resistant comb methods use about half of the storage for pre-computed points as HPB's comb method, yet save $2^{w-1} - w$ point additions in the precomputation stage. Our Algs. 8 and 9 have the same time cost as HPB's method in the evaluation stage, while our Alg. 6 saves one point

Table 3: **Space and average time costs for SPA-nonresistant signed m-ary method and SPA-resistant Möller's and OT's window methods ($n = \lceil \log_2 \rho \rceil$)**

	Signed m-ary window	Möller's window[a]	OT's window
d	$\lceil \frac{n}{w} \rceil$	$\lceil \frac{n}{w} \rceil$	$\lceil \frac{n}{w} \rceil$
Storage	2^{w-2}	$2^{w-1}+1$	2^{w-1}
Pre-Stage	$1D$ $(2^{w-2}-1)A$	$(2^{w-2}+1)D$ $(2^{w-2}-1)A$	$1D$ $(2^{w-1}-1)A$
Eva-Stage	$(wd-w)D$ $(\frac{wd+1}{w+1}-1)A$	$(wd-w)D$ $(d-1)A$	$(wd-w+1)D$ dA
Total Cost	$(wd-w+1)D$ $(2^{w-2}+\frac{wd+1}{w+1}-2)A$	$(2^{w-2}+wd-w+1)D$ $(2^{w-2}+d-2)A$	$(wd-w+2)D$ $(2^{w-1}+d-1)A$

.

[a]Replacing d by d+1 for Möller's window method if the case $d' = d + 1$ occurs.

doubling and one point addition in this stage. From the data in Table 2, our Alg. 6 is the most efficient SPA-resistant comb method if n is not divisible by w. When n is divisible by w, our Alg. 9 is recommended since $\lceil \frac{n+1}{w} \rceil$ is one larger than $\frac{n}{w}$ in this case, resulting in higher time cost.

We would also like to compare our SPA-resistant methods with other efficient SPA-resistant point multiplication methods. The space and time costs for Möller's window method [18] and OT's window method [21] are listed in the second and third columns of Table 3. From Tables 2 and 3, our SPA-resistant comb methods store the same number of pre-computed points as OT's window method and one point less than Möller's window method. Since point doubling is faster than point addition, Möller's method is the most efficient SPA-resistant method if the total time cost including both precomputation and evaluations stages is considered. OT's window method is also more efficient than ours in this case. If precomputation points can be calculated in advance or elsewhere, i.e., if only the time cost in the evaluation stage is considered, our comb methods are much faster than both Möller's method and OT's method. Our SPA-resistant comb methods are always better than HPB's comb method no matter if the precomputation is included or not in the time cost estimation.

To make a point multiplication method resistant to SPA attacks, additional operations are needed to remove dependency of the cryptographic execution procedure on the specific value of k.

This means that efficiency of an SPA-resistant method is lowered in general as compared to SPA-nonresistant methods with a similar approach. The most efficient point multiplication method if total time cost including both precomputation and evaluation stages is considered is the *signed m-ary window method* ([4], p. 70), an SPA-nonresistant method. The space and time costs for the signed m-ary window method are listed in the first column of Table 3. This method requires less points to be stored yet runs faster than all the SPA-resistant methods, ours or others. The space and time penalty is well-deserved in many applications that security is a top priority. If only the evaluation stage is considered, our comb methods are faster than the signed m-ary window method. In fact, our comb methods are the fastest in this case.

5 Application Examples

Our comb methods can be used in many ECC applications. They are particularly efficient if precomputation can be computed in advance or by somebody else, which is the case in many applications. This section describes a couple of such application scenarios. One example is a system that a smart card is used to provide a tamper-resistant and secure environment to store secrets and execute critical cryptographic operations, while a separate, more powerful computing subsystem is responsible for the rest operations. Cellular phones and wireless application protocol (WAP) devices are typical examples of such a system. In a cellular phone, the Subscriber Identification Module (SIM) card is a smart card to store securely critical subscriber's information and authentication and encryption algorithms responsible for providing legitimate access to the wireless network. The phone's CPU, memory, and storage are responsible to the rest operations. A Wireless Identity Module (WIM) card play a similar role in a WAP device. In such a system, we may be able to dedicate the precomputation to the more powerful device's CPU while use the smart card to execute the evaluation stage, if the computed points by the device's CPU are observable but not tamperable. Note that precomputation does not contain any secrets unless the point itself is a secret.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is another example. ECDSA is the elliptic curve analogue of the Digital Signature Algorithm (DSA) specified in a U. S. government standard called the Digital Signature Standard [27]. ECDSA has been accepted in many standards [29]. It includes signature generation and verification. Let P be a publicly known elliptic curve point and ρ be the prime order of the point P. A signature is generated and verified in the following way:

ECDSA Signature Generation: To sign a message m, an entity A associated with a key pair (d, Q) executes the following steps:

1. Select a random or pseudorandom integer $k, 1 \leqslant k < \rho$.
2. Compute $kP = (x_1, y_1)$ and $r = x_1 \mod \rho$. If $r = 0$, go to Step 1.
3. Compute $k^{-1} \mod \rho$.
4. Compute $e =$ SHA-1(m), where SHA-1 is a Secure Hash Algorithm (SHA) specified in the Secure Hash Standard [28].
5. Compute $s = k^{-1}(e + dr) \mod \rho$. If $s = 0$, go to Step 1.
6. The signature generated by A for the message m is (r, s).

ECDSA Signature Verification: To verify A's signature (r, s) on a message m, an entity B obtains A's public key Q and executes the following steps:

1. Verify that r and s are integers in the interval $[1, \rho - 1]$.
2. Compute $e =$ SHA-1(m).

3. Compute $w = s^{-1} \mod \rho$.

4. Compute $u_1 = ew \mod \rho$ and $u_2 = rw \mod \rho$.

5. Compute $X = u_1P + u_2Q = (x_1, y_1)$. If $X = \mathcal{O}$, reject the signature. Otherwise, compute $v = x_1 \mod \rho$.

6. Accept the signature if and only if $v = r$.

For an ECDSA signature, P, ρ and r, s, e are public values. The scalar k has to be kept secret. Otherwise the private key d can be derived from the equation $s = k^{-1}(e + dr) \mod \rho$. Therefore the private key d and the ECDSA signature generation must be securely stored and executed. This can be conveniently achieved with a smart card, which stores P's precomputed points and uses a point multiplication method resistant to power analysis to calculate kP. Our SPA-resistant comb methods are ideal for this application since only the evaluation stage is executed in generating a signature. ECDSA signature verification, on the other hand, does not use any secret key. In verifying an ECDSA signature, our SPA-nonresistant comb methods can be used to compute u_1P and the signed m-ary window method is used to compute u_2Q. This is an efficient combination of point multiplication methods since P's precomputation points can be calculated in advance, and a comb method is more efficient than other methods in this case. As a contrast, Q's precomputation points cannot be calculated in advance since the public key Q varies from one entity to another. The signed m-ary window method is appropriate in this case.

6 Conclusion

In this paper, we presented a novel signed odd-only comb recoding algorithm to convert a scalar k to a sequence of signed odd-only nonzero comb bit-columns. Using this recoding algorithm, we presented several novel SPA-nonresistant and SPA-resistant comb methods to calculate point multiplication for elliptic curve cryptosystems. Our comb methods store less number of points and runs faster than the original SPA-nonresistant comb method and HPB's SPA-resistant comb method. In addition, our proposed comb methods inherit a comb method's advantage – running much faster than any window methods, such as SPA-resistant Möller's and OT's window methods, and SPA-nonresistant signed m-ary window method, when precomputation points are calculated in advance or elsewhere. These features of our comb methods are very desirable in smart cards and embedded systems where power and computing resources are at a premium. When combined with randomization techniques and certain precautions in selecting elliptic curves and parameters, our presented SPA-resistant comb methods can thwart all side-channel attacks.

References

[1] V. S. Miller, "Use of Elliptic Curves in Cryptography," *Advances in Cryptology – CRYPTO'85*, H. C. Williams, Ed., LNCS 218, pp. 417–426, Springer-Verlag, 1986.

[2] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203–209, 1987.

[3] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. of ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[4] I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography*, Cambridge Univ. Press, 1999.

[5] C. Lim and P. Lee, "More Flexible Exponentiation with Precomputation," *Advances in Cryptology – CRYPTO'94,* Y. G. Desmedt, Ed., LNCS 839, pp. 95–107, Springer-Verlag, 1994.

[6] P. C. Kocher, "Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS and Other Systems," *Advances in Cryptology – CRYPTO'96*, N. Koblitz, Ed., LNCS 1109, pp. 104–113, Springer-Verlag, 1996.

[7] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Advances in Cryptology – CRYPTO'99*, M. Wiener, Ed., LNCS 1666, pp. 388–397, Springer-Verlag, 1999.

[8] J. S. Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems," *Cryptographic Hardware and Embedded Systems – CHES'99,* Ç. K. Koç and C. Paar Eds., LNCS 1717, pp. 292–302, Springer-Verlag, 1999.

[9] T. S. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," *Cryptographic Hardware and Embedded Systems – CHES'2000*, Ç. K. Koç and C. Paar, Eds., LNCS 1965, pp. 238–251, Springer-Verlag, 2000.

[10] K. Okeya and K. Sakurai, "A Second-Order DPA Attack Breaks a Window-Method Based Countermeasure against Side Channel Attacks," *Proc. 5th Intl. Conf. on Information Security*, LNCS 2433, pp. 389–401, Springer-Verlag, 2002.

[11] L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems," *Public Key Cryptography – PKC'2003*, Y. G. Desmedt Ed., LNCS 2567, pp 199–211, Springer-Verlag, 2003.

[12] P. V. Liardet and N. P. Smart, "Preventing SPA/DPA in ECC Systems Using the Jacobi Form," *Cryptographic Hardware and Embedded Systems – CHES'2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds., LNCS 2162, pp. 391–401, Springer-Verlag, 2001.

[13] M. Joye and J. J. Quisquater, "Hessian Elliptic Curves and Side-Channel Attacks," *Cryptographic Hardware and Embedded Systems – CHES'2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds., LNCS 2162, pp. 402–410, Springer-Verlag, 2001.

[14] National Institute of Standards and Technology (NIST), *Digital Signature Standard (DSS)*, FIPS Pub. 186-2, 2000.

[15] K. Okeya, H. Kurumatani, and K. Sakurai, “Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications,” *Public Key Cryptography– PKC'2000*, H. Imai, Y. Zheng, Eds., LNCS 1751, pp. 238–257, Springer-Verlag, 2000.

[16] P. L. Montgomery, “Speeding the Pollard and Elliptic Curve Methods of Factorization,” *Mathematics of Computation*, vol. 48, pp. 243–264, 1987.

[17] B. Möller, “Securing Elliptic Curve Point Multiplication against Side-Channel Attacks,” *Information Security,* G.I Davida and Y. Frankel, Eds., LNCS 2200, pp. 324–334, Springer-Verlag, 2001.

[18] B. Möller, “Securing Elliptic Curve Point Multiplication against Side-Channel Attacks, Addendum: Efficiency Improvement,” http://www.informatik.tudarmstadt.de/TI/Mitarbeiter/moeller/ecc-scaisc01.pdf, 2001.

[19] M. Hedabou, P. Pinel, and L. Bébéteau, “A Comb Method to Render ECC Resistant against Side Chiannel Attacks,” http://eprint.iacr.org/2004/342.pdf, 2004.

[20] M. Joye and C. Tymen, “Protections against Differential Analysis for Elliptic Curve Cryptography – An Algebraic Approach,” *Cryptographic Hardware and Embedded Systems – CHES'2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds., LNCS 2162, pp. 377–390, Springer-Verlag, 2001.

[21] K. Okeya and T. Takagi, “The Width-w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplication Secure against Side Channel Attacks,” *Topics in Cryptology, The Cryptographers' Track at the RSA Conference 2003 (CT-RSA 2003)*, LNCS 2612, pp. 328–342, 2003.

[22] K.Okeya and T.Takagi, “A More Flexible Countermeasure against Side Channel Attacks Using Window Method” *cryptographic Hardware and Embedded Systems – CHES’2003*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds., LNCS 2779, pp. 397–410, 2003.

[23] W. Fischer, C. Giraud, E. W. Knudsen, and J.-P. Seifert, “Parallel Scalar Multiplication on General Elliptic Curve over F_p Hedged against Non-Differential Side-Channel Attacks,” *IACR, Cryptography ePrint Archieve 2002/007*, http://eprint.iacr.org/2002/007, 2002.

[24] E. Brier and M. Joye, “Weierstrass Elliptic Curves and Side-Channel Attacks,” *Public Key Cryptography (PKC2002)*, LNCS 2274, pp. 335–345, 2002.

[25] T. Izu, and T. Takagi, “A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks,” *Public Key Cryptography (PKC 2002)*, LNCS 2274, pp. 280–296, 2002.

[26] B. Chevallier-Mames, M. Ciet, and M. Joye, “Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity,” *IEEE Transaction on Computers*, vol. 53, no. 6, pp. 760–768, June 2004.

[27] National Institute of Standards and Technology, *Digital Signature Standard*, FIPS Pub. 186, 1994.

[28] National Institute of Standards and Technology, *Secure Hash Standard*, FIPS Pub. 180-1, 1993.

[29] D. Johnson and A. Menezes, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” *Technical Report CORR 99-34*, Dept. of C&O, University of Waterloo, Canada. Also available from http://www.cacr.math.uwaterloo.ca.