Parallelized Side-Channel Attack Resisted Scalar Multiplication Using q-Based Addition-Subtraction k-chains

Kittiphop Phalakarn
Department of Computer Engineering
Chulalongkorn University
email: kittiphop.ph@student.chula.ac.th

Kittiphon Phalakarn
Department of Computer Engineering
Chulalongkorn University
email: kittiphon.p@student.chula.ac.th

Vorapong Suppakitpaisarn
Department of Computer Science
The University of Tokyo
email: vorapong@is.s.u-tokyo.ac.jp

***Abstract*—This paper presents parallel scalar multiplication techniques for elliptic curve cryptography using q-based addition-subtraction k-chain which can also effectively resist side-channel attack. Many techniques have been discussed to improve scalar multiplication, for example, double-and-add, NAF, w-NAF, addition chain and addition-subtraction chain. However, these techniques cannot resist side-channel attack. Montgomery ladder, random w-NAF and uniform operation techniques are also widely used to prevent side-channel attack, but their operations are not efficient enough comparing to those with no side-channel attack prevention. We have found a new way to use k-chain for this purpose. In this paper, we extend the definition of k-chain to q-based addition-subtraction k-chain and modify an algorithm proposed by Jarvinen et al. to generate the q-based addition-subtraction k-chain. We show the upper and lower bounds of its length which lead to the computation time using the new chain techniques. The chain techniques are used to reduce the cost of scalar multiplication in parallel ways. Comparing to w-NAF, which is faster than double-and-add and Montgomery ladder technique, the maximum computation time of our q-based addition-subtraction k-chain techniques can have up to 25.92% less addition costs using only 3 parallel computing cores. We also discuss on the optimization for multiple operand point addition using hybrid-double multiplier which is proposed by Azarderakhsh and Reyhani-Masoleh. The proposed parallel chain techniques can also tolerate side-channel attack efficiently.**[1]

keywords*—*Information and Communication Security; Efficient Implementations; Parallel Algorithms; Elliptic Curve Cryptography; Scalar Multiplication; k-Chain; Side-Channel Attack Countermeasure

I. Introduction

Today's cryptosystem requires faster computation for more security. The cost of scalar multiplication is very important for elliptic curve cryptography. Many works try to decrease this calculation cost, for example, double-and-add technique (cf. [1]) which uses simple binary representation. It can be calculated faster using NAF [2] and w-NAF [3] technique. NAF technique uses binary representation with digit set $\{-1, 0, 1\}$ and w-NAF representation uses larger digit set. Another technique used today is wr-NAF [4] which generalizes w-NAF using base r representation.

There is another kind of techniques which is different from double-and-add. Addition chain (cf. [5]) and addition-subtraction chain (a/s chain) [6] techniques are used for the same purpose as the general cases for double-and-add and NAF [7], [8]. There are many variations on the addition chain and a/s chain such as addition-multiplication chain [9], q-addition chain [10] and k-chain [11]. Currently, k-chain is only used to improve finite field element inversion [11] and we are interested in using this k-chain to improve scalar multiplication.

One important problem for scalar multiplication is side-channel attack. The power used in each calculation step can be measured and give information on which type of operation is being calculated. This is called 'power attack' [12]. Time of each computation step can also be measured for the same purpose. The latter attack is called 'timing attack' [13]. The sequences of the monitored operations can tell which scalar has been used to multiply. The multiplication techniques mentioned earlier are not tolerate side-channel attack. Some studies try to add some random dummy operations to hide the real power and time used, but these operations increase the calculation costs which are now close to the worst case of each technique. Montgomery ladder [14], random w-NAF [15] and uniform operation techniques [16] are widely used to prevent side-channel attack, but their operations are not efficient enough comparing to those with no side-channel attack prevention.

A. Contribution of this paper

To efficiently resist the side-channel attack, we have found a new way to use k-chain for scalar multiplication. In this paper, we generalize the definition of k-chain to addition-subtraction k-chain (a/s k-chain) and q-based addition-subtraction k-chain (q-based a/s k-chain). We modify the algorithm from [11] to generate the q-based a/s k-chain. From the algorithms, we found that summation of all digits in the number representation affects the length of the chain. This summation is called 'relaxed cost' [17]. We use the relaxed reduced representation's properties from [17] to prove the theorem of the maximum relaxed cost of the representation. This leads to the upper and lower bounds of the chain's length and the worst case of the computation time for scalar multiplication using q-based a/s k-chain technique.

We propose a parallel technique of 2-based a/s 4-chain and generalize to q-based a/s k-chain for any positive integers q and k greater than 2. We also propose an optimization technique for multiple operand point addition for twisted

[1]This paper has been accepted for publication at proceedings of the Fourth International Symposium on Computing and Networking (CANDAR 2016), which is published by IEEE. It has been further edited by IEEE, and the final version is appearing at http://ieeexplore.ieee.org/document/7818605/.

Edwards curves [18] by using hybrid-double multiplier [19], [20].

We compare each technique using worst case computation time due to the properties of side-channel attack. Comparing to the best w-NAF (with $w = 5$) [4], which is faster than double-and-add and Montgomery ladder technique, the maximum computation time of our q-based addition-subtraction k-chain technique using 128-bit scalar is slightly smaller with 2 computing cores. For large scalar, our chain technique can have up to 25.92% less additions using only 3 parallel computing cores, but if using 2 computing cores, our technique can have up to 11.11% more additions.

Finally, we discuss on the appropriate value of k in q-based a/s k-chain and have some security analysis of the proposed chain techniques on side-channel attack.

B. Paper Organization

The paper is structured as follows: In Section II, we present some previous works on this topic, including relaxed reduced representation, k-chain, double-and-add technique, w-NAF technique and how to calculate the cost of scalar multiplication. In Section III, we give definitions on a/s k-chain and q-based a/s k-chain. We also modify the k-chain algorithm to generate q-based a/s k-chain. In Section IV, we prove the maximum relaxed cost of signed digit representations and propose a theorem on the upper and lower bounds of q-based a/s k-chain length. This theorem leads us to the computation time using the chain techniques. In Section V, we propose some parallel techniques and give an optimization on multiple operand point addition using hybrid-double multiplier [19], [20]. We compare the costs of our techniques in Section VI and compare to other well-known techniques, w-NAF and Montgomery ladder. In the same section, we also discuss on the value of k and give some security analysis of the proposed chain techniques on side-channel attack. Finally, we conclude our paper in Section VII.

II. Preliminaries

In this section, we present some previous works, including relaxed reduced representation [17], k-chain [11], double-and-add technique (cf. [1]) and w-NAF technique [3] for scalar multiplication.

A. Relaxed Reduced Representation

We use the definitions and theorems from [17], and we will focus on relaxed cost because they are more compatible with our applications.

Definition 1 (set of all representations [17]). *The set of all representations of an integer n in base $q \geq 2$ is defined as*

$$R_q(n) := \left\{ \varepsilon = \langle \varepsilon_0, ..., \varepsilon_\ell \rangle_q \;\middle|\; \ell \in \mathbb{N}, \varepsilon_i \in \mathbb{Z}, \varepsilon_\ell \neq 0, n = \sum_{i=0}^{\ell} \varepsilon_i q^i \right\}$$

and for each $\varepsilon \in R_q(n)$, ℓ is called the length of the representation ε and written as $|\varepsilon|$.

Example 2. *Given $n = 15$ and $q = 4$, we have some representations $\langle 3, 3\rangle_4, \langle -1, 0, 1\rangle_4, \langle 11, 1\rangle_4 \in R_4(15)$ with length 1, 2 and 1, respectively.*

Definition 3 (relaxed cost of the representation [17]). *The relaxed cost of representation $\varepsilon \in R_q(n)$ is defined as*

$$c'(\varepsilon) = c'\left(\langle \varepsilon_0, ..., \varepsilon_\ell \rangle_q\right) := \sum_{i=0}^{\ell} |\varepsilon_i|.$$

Definition 4 (relaxed reduced representation). *The representation $\varepsilon^* \in R_q(n)$ is relaxed reduced if*

$$\varepsilon^* \in \operatorname*{arg\,min}_{\varepsilon \in R_q(n)} c'(\varepsilon).$$

Example 5. *From Example 2, the relaxed costs of the example representations are $6, 2$ and 12, respectively. It is easy to see that $\langle -1, 0, 1\rangle_4$ is relaxed reduced representation of $R_4(15)$.*

Definition 6 (reduction rules [17]). *Given $\varepsilon \in R_q(n)$, $f : R_q(n) \to R_q(n)$ is defined as follows. The rules are applied in this order at the first occurrence from the left side until ε first changes and return the result. If there is no match, $f(\varepsilon) = \varepsilon$.*

For $|\eta_0| > \frac{q}{2}, \eta_1 \in \mathbb{Z}$, and $u := \left\lceil \left| \frac{\eta_0}{2(q-1)} \right| \right\rceil$:

$$\langle \eta_0, \eta_1 \rangle_q \to \langle \eta_0 - uq \; sign(\eta_0), \eta_1 + u \; sign(\eta_0) \rangle_q.$$

For $\eta_0 < 0$ and $2 \mid q$:

$$\left\langle \frac{q}{2}, \eta_0 \right\rangle_q \to \left\langle -\frac{q}{2}, \eta_0 + 1 \right\rangle_q.$$

For $\eta_0 > 0$ and $2 \mid q$:

$$\left\langle -\frac{q}{2}, \eta_0 \right\rangle_q \to \left\langle \frac{q}{2}, \eta_0 - 1 \right\rangle_q.$$

For $\eta_0 < \frac{q}{2}$ and $2 \mid q$:

$$\left\langle \frac{q}{2}, \frac{q}{2}, \eta_0 \right\rangle_q \to \left\langle -\frac{q}{2}, -\frac{q}{2} + 1, \eta_0 + 1 \right\rangle_q.$$

For $-\frac{q}{2} < \eta_0$ and $2 \mid q$:

$$\left\langle -\frac{q}{2}, -\frac{q}{2}, \eta_0 \right\rangle_q \to \left\langle \frac{q}{2}, \frac{q}{2} - 1, \eta_0 - 1 \right\rangle_q.$$

Theorem 7 ([17]). *The reduction rules can be applied to the representation $\varepsilon \in R_q(n)$ many times until there is no change. The representation ε is relaxed reduced if and only if $f(\varepsilon) = \varepsilon$.*

Theorem 8 (Average Relaxed Cost of Relaxed Reduced Representation [17]). *Given positive integer m and base $q \geq 2$, the average relaxed cost of the relaxed reduced representations of the integer $0, ..., q^m - 1$ is*

$$\sum_{i=0}^{q^m-1} \frac{c'(\varepsilon^* \in R_q(i))}{q^m} = \begin{cases} \left(\frac{q}{4} - \frac{1}{4q}\right) m + O(1) & \text{if } 2 \nmid q, \\ \left(\frac{q}{4} - \frac{1}{2(q+1)}\right) m + O(1) & \text{if } 2 \mid q. \end{cases}$$

Algorithm 1 has been proposed in [17] to generate relaxed reduced representation of an integer.

Algorithm 1 [17] generating base q relaxed reduced representation of positive integer n

```
Input: $n > 0$, $q \geq 2$
Output: relaxed reduced representation $\varepsilon^* \in R_q(n)$
1: $\varepsilon^* \leftarrow \langle\ \rangle_q$
2: $m \leftarrow n$
3: while $m > 0$ do
4:     $a \leftarrow (m \bmod q)$
5:     if $a > q/2$ or $(a = q/2$ and $m \bmod q^2 \geq q^2/2)$ then
6:         $a \leftarrow a - q$
7:     end if
8:     $m \leftarrow (m - a)/q$
9:     concatenate $a$ to the right end of $\varepsilon^*$
10: end while
11: return $\varepsilon^*$
```

To generate reduced representation (the relaxed reduced representation with the shortest length), change the condition in the if statement in line 5 to "not $(a < q/2$ or $(a = q/2$ and $m \bmod q^2 < q^2/2)$ or $m = (q+1)/2$ or $m = q/2 + q^2/2)$".

B. k-Chain

We give the definition of k-chain from [11]. Previous works on addition chain, a/s chain and k-chain can be found in [5], [6], [11].

Definition 9 (k-chain [11]). *Given positive integer $k \geq 2$ and n, a k-chain for n is defined as*

$$v = [v(0), ..., v(s)]_k$$

with $v(0) = 1$, $v(s) = n$, and $v(i) = \sum_{h=0}^{k-1} v(i_h)$ for all $0 < i \leq s$ and $-1 \leq i_h < i$ with $v(-1) = 0$. The length of the chain is equal to s.

The algorithm for generating k-chain from traditional base k representation is given in [11] and we show it below. The same paper also proved the upper and lower bounds of the length of the optimal k-chain by using the given algorithm. We state it in Theorem 10.

Remark: At line 3 of Algorithm 2, the length of the output k-chain will be the same regardless to the order of x_i being chosen.

Theorem 10 (bound of k-chain's length [11]). *Given positive integers $k \geq 2$ and n, The upper and lower bounds of the optimal length s of k-chain for n are*

$$\lceil \log_k n \rceil \leq s \leq \lfloor \log_k n \rfloor + \left\lceil \frac{c'(\varepsilon) - 1}{k - 1} \right\rceil$$

where $\varepsilon \in R_k(n)$ is a traditional base k representation, i.e., using digit set $\{0, 1, ..., k-1\}$.

C. Current Techniques on Scalar Multiplication

To calculate nP from the given integer n and elliptic curve point P, the most simple technique used is double-and-add (cf. [1]). Assume that we have the traditional binary representation

Algorithm 2 [11] Algorithm for generating k-chain for integer n

```
Input: $k$, $n = \sum_{i=0}^{\ell} n_i k^i = \langle n_0, ..., n_\ell \rangle_k$ (traditional representation)
Output: $k$-chain for $n$
1: $v(i) \leftarrow k^i$ for $0 \leq i \leq \ell$
2: $j \leftarrow 0$
3: for all $x_i \neq 0$ in $x = \{n_0, n_1, ..., n_{\ell-1}, n_\ell - 1\}$ do
4:     $(t_j, t_{j+1}, ..., t_{j+|x_i|-1}) \leftarrow (i, ...i)$
5:     $j \leftarrow j + |x_i|$
6: end for
7: $(t_j, t_{j+1}, ..., t_{\lceil j/(k-1) \rceil (k-1) - 1}) \leftarrow (-1, -1, ..., -1)$
8: for $i = 0$ to $\lfloor j/(k-1) \rfloor$ do
9:     $v(\ell + i + 1) \leftarrow v(\ell + i) + v(t_{i(k+1)}) + v(t_{i(k+1)+1})$
10:                        $+ \cdots + v(t_{i(k+1)+k-2})$
11: end for
12: return $v$
```

of n, $\langle n_0, ..., n_\ell \rangle_2$, we can compute nP from $2(\cdots 2(2(n_\ell P) + n_{\ell-1}P) + \cdots) + n_0 P$

Example 11. *The binary representation of 10 is $\langle 0, 1, 0, 1 \rangle_2$. Thus, we can calculate $10P$ from $2(2(2(\boldsymbol{1}P) + \boldsymbol{0}P) + \boldsymbol{1}P) + \boldsymbol{0}P$.*

To see the calculation cost of this technique, we define the cost of point doubling as $\mathcal{D}$ and point addition as $\mathcal{A}$. Given positive integer n and traditional binary representation $\varepsilon \in R_2(n)$, the cost for scalar multiplication with double-and-add technique is defined as

$$Cost_{n,double} = \mathcal{D} \times |\varepsilon| + \mathcal{A} \times (c'(\varepsilon) - 1).$$

Remark: For twisted Edwards curves [18], which is the current fastest curve, the cost for point addition $\mathcal{A}$ is now recorded at $11\mathcal{M}$ where $\mathcal{M}$ is the cost for field elements multiplication. The cost for point doubling $\mathcal{D}$ is recorded at $7\mathcal{M}$.

Given n as an m-bit integer where $m := \lfloor \log_2 n \rfloor + 1$, We define average and maximum costs for scalar multiplication as follows.

$$Avg_{m,technique} = \frac{\sum_{i=0}^{2^m - 1} Cost_{i,technique}}{2^m}$$
$$Max_{m,technique} = \max_{0 \leq i \leq 2^m - 1} Cost_{i,technique}$$

Given a positive integer n with m bits, the length of its binary representation is exactly m. From Subsection II-A, we have average relaxed cost equals to $\frac{1}{2}m + O(1)$ and it is easy to see that the maximum relaxed cost equals to $m + O(1)$. Thus, the final average and maximum costs for double-and-add technique are

$$Avg_{m,double} = \mathcal{D} \times m + \mathcal{A} \times \frac{1}{2}m + O(1),$$
$$Max_{m,double} = \mathcal{D} \times m + \mathcal{A} \times m + O(1).$$

Another technique widely used is w-NAF technique. The w-NAF technique was introduced in [3] and generalized to wr-NAF in [4]. The paper stated that given the window size

w and base r, the nonzero density of the base r representation is $\frac{r-1}{w(r-1)+1}$ and the number of points that have to be pre-computed is $\frac{r^w - r^{w-1}-2}{2}$. It is easy to see that the worst case for nonzero density is $\frac{1}{w}$. With wr-NAF, we use number of nonzero digits instead of relaxed cost. To calculate nP, we define $m := \lfloor \log_2 n \rfloor + 1$. The average and maximum costs for wr-NAF technique are similar to those for double-and-add and can be written as follows.

$$\begin{aligned} Avg_{m,wr\text{-NAF}} &= \mathcal{R} \times \frac{m}{\log_2 r} \\ &\quad + \mathcal{A} \times \left(\frac{r-1}{w(r-1)+1} \left(\frac{m}{\log_2 r} \right) \right. \\ &\qquad \left. + \frac{r^w - r^{w-1} - 2}{2} \right), \end{aligned}$$

$$\begin{aligned} Max_{m,wr\text{-NAF}} &= \mathcal{R} \times \frac{m}{\log_2 r} + \mathcal{A} \times \left(\frac{1}{w} \left(\frac{m}{\log_2 r} \right) \right. \\ &\qquad \left. + \frac{r^w - r^{w-1} - 2}{2} \right), \end{aligned}$$

when $\mathcal{R}$ is the calculation cost for multiplying the point by r.

III. q-Based Addition-Subtraction k-Chain

We generalize the definition of k-chain to a/s k-chain and we will extend the definition to q-based a/s k-chain.

Definition 12 (addition-subtraction k-chain)**.** *Given positive integer $k \geq 2$ and n, an addition-subtraction k-chain (a/s k-chain) for n is defined as*

$$v = [v(0), ..., v(s)]_k$$

with $v(0) = 1$, $v(s) = n$, and $v(i) = \sum_{h=0}^{k-1} \pm v(i_h)$ for all $0 < i \leq s$ and $-1 \leq i_h < i$ with $v(-1) = 0$.

Example 13. *An a/s 3-chain for 8 is $[1, 3, 9, 8]_3$ because $3 = v(1)+v(1)+v(1) = 1+1+1$, $9 = v(2)+v(2)+v(2) = 3+3+3$ and $8 = v(3) - v(1) + v(-1) = 9 - 1 + 0$. Other examples are $[1, 3, 7, 8]_3$ and $[1, 2, 6, 10, 8]_3$.*

Definition 14 (q-based addition-subtraction k-chain)**.** *An addition-subtraction k-chain v is a q-based addition-subtraction k-chain (q-based a/s k-chain), if and only if, $q \leq k$ and there exists an integer j such that for all $0 < i \leq j$, $v(i) = q^i$. The $[v(0), ..., v(j)]_k$ part is called "power part" and $[v(j+1), ..., v(s)]_k$ part is called "addition part".*

Example 15. *A 2-based a/s 3-chain for 10 is $[1, 2, 4, 8, 10]_3$. The power part is $[1, 2, 4, 8]_3 = [2^0, 2^1, 2^2, 2^3]_3$ and the addition part is $[10]_3$ where $10 = v(3) + v(1) + v(-1) = 8+2+0$. Other examples are $[1, 2, 4, 10]_3$, $[1, 2, 3, 9, 10]_3$ and $[1, 2, 4, 8, 16, 10]_3$. You should notice that $[1, 3, 9, 10]_3$ is an a/s 3-chain but not a 2-based a/s 3-chain.*

Next, we modify Algorithm 2 to Algorithm 3 for generating q-based a/s k-chain. If the desired output is just a/s k-chain, you can use $q = k$.

Algorithm 3 Modified algorithm for generating q-based addition-subtraction k-chain for an integer n

```
Input: $q$, $k$, $n = \sum_{i=0}^{\ell} n_i q^i = \langle n_0, ..., n_\ell \rangle_q$
Output: $q$-based addition-subtraction $k$-chain for $n$
1: $v(i) \leftarrow q^i$ for $0 \leq i \leq \ell$
2: $j \leftarrow 0$
3: for all $x_i \neq 0$ in $x = \{n_0, n_1, ..., n_{\ell-1}, n_\ell - 1\}$ do
4:     $(t_j, t_{j+1}, ..., t_{j+|x_i|-1}) \leftarrow ((i+1)\ \mathrm{sign}(x_i), ...,$
5:                                      $(i+1)\ \mathrm{sign}(x_i))$
6:     $j \leftarrow j + |x_i|$
7: end for
8: $(t_j, t_{j+1}, ..., t_{\lceil j/(k-1) \rceil (k-1)-1}) \leftarrow (0, 0, ..., 0)$
9: for $i = 0$ to $\lfloor j/(k-1) \rfloor$ do
10:     $v(\ell+i+1) \leftarrow v(\ell+i) + v(t_{i(k+1)} - 1)\ \mathrm{sign}(t_{i(k+1)})$
11:                      $+v(t_{i(k+1)+1} - 1)\ \mathrm{sign}(t_{i(k+1)+1})$
12:                      $+\cdots$
13:                      $+v(t_{i(k+1)+k-2} - 1)\ \mathrm{sign}(t_{i(k+1)+k-2})$
14: end for
15: return $v$
```

There are some differences between Algorithm 2 and Algorithm 3. In Algorithm 3, the input is a base q signed digit representation instead of traditional representation. We will use relaxed reduced representation as an input here. In line 3-5 and 10-13, we use sign function to separate addition and subtraction operations. We have to use $+1$ in line 3-5, use 0 in line 8 and use -1 on line 10-13 to make the index values not equal to zero.

Example 16. *We will follow Algorithm 3 to generate 4-based a/s 5-chain for $n = 1977$. From Algorithm 1, the relaxed reduced representation in base 4 for 1977 is $\langle 1, -2, 0, -1, 0, 2 \rangle_4$ and we will use it as an input. Following Algorithm 3, in line 1, we will have $v = [1, 4, 16, 64, 256, 1024]_5$. By choosing x_i from left to right, at the end of line 7 we will have $t = (1, -2, -2, -4, 6, 0, 0, 0)$. In line 9 to 14, the next element in the chain is calculated from the first 4 elements of t, $v(6) = v(5) + v(0) - v(1) - v(1) - v(3) = 953$. Then, the last element is calculated from the last 4 elements of t, $v(7) = v(6) + v(5) + v(-1) + v(-1) + v(-1) = 1977$. The algorithm returns the 4-based a/s 5-chain for 1977, $[1, 4, 16, 64, 256, 1024, 953, 1977]_5$. We show the flow of the algorithm in Fig. 1.*

From Algorithms 2 and 3, you can see that relaxed cost of the representation affects the length of the chain. In the next section, we will discuss on the relaxed cost, the bound of the length of the chain and show proofs on this. These will lead to the computation time of scalar multiplication using q-based a/s k-chain technique.

IV. Worst Case of Computation Time Using q-Based Addition-Subtraction k-Chain Technique

Before showing the computation time, we give some discussions on the relaxed reduced representations and its maximum

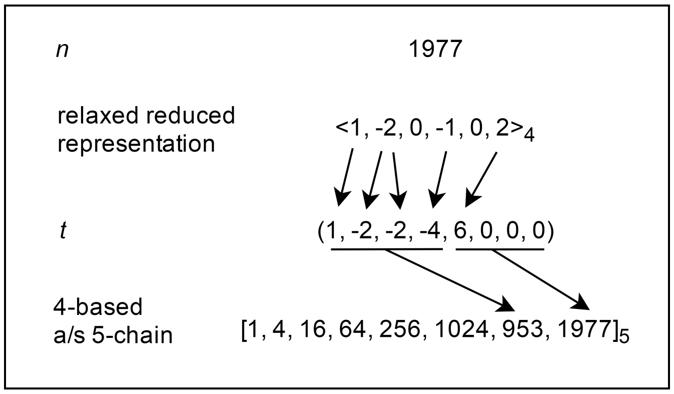

Fig. 1. Generating 4-based a/s 5-chain of 1977 using Algorithm 3

relaxed costs. We will use these relaxed reduced representations as inputs for Algorithm 3. They also directly affect the length of our q-based a/s k-chain and the computation time.

A. Worst Case of Relaxed Cost of Relaxed Reduced Representation

Lemmas and theorems discussed in Subsection II-A show many properties of the relaxed reduced representations and we will use these properties to prove our works. We then conclude the worst case of relaxed cost of relaxed reduced representation in Theorem 21.

Lemma 17. *For all natural number m and odd base $q \geq 3$, there exists an integer $0 \leq w \leq q^m - 1$ with its relaxed reduced representation $\varepsilon^* \in R_q(w)$ such that $c'(\varepsilon^*) = 1 + \left(\frac{q-1}{2}\right)(m)$.*

Proof. Let $b := \frac{q-1}{2}$ and $w := \frac{q^m+1}{2}$. The relaxed reduced representation of w is $\varepsilon^* = \langle \varepsilon^*_0, ..., \varepsilon^*_m \rangle_q = \langle -b, ..., -b, 1 \rangle_q \in R_q(w)$. You can see that $0 \leq w = \sum_{i=0}^{m} \varepsilon^*_i q^i \leq q^m - 1$ and $c'(\varepsilon^*) = 1 + bm = 1 + \left(\frac{q-1}{2}\right)(m)$. ■

Lemma 18. *For all natural number m and odd base $q \geq 3$, $c'(\varepsilon^*)$ is at most $1 + \left(\frac{q-1}{2}\right)(m)$ for relaxed reduced representation $\varepsilon^* \in R_q(w)$, $0 \leq w \leq q^m - 1$.*

Proof. Let $b := \frac{q-1}{2}$. We assume that there exists an integer $0 \leq w' \leq q^m - 1$ with a relaxed reduced representation $\varepsilon'^* = \langle \varepsilon'^*_0, ..., \varepsilon'^*_k \rangle_q \in R_q(w')$ such that $c'(\varepsilon'^*) > 1 + \left(\frac{q-1}{2}\right)(m) = 1 + bm$.

Case the length $k > m$, from relaxed reduced representation's properties in [1] that $|\varepsilon'^*_i| \leq b$, we can derive the inequalities as follows. From the assumption, we know that $\varepsilon'^*_k > 0$.

$$\begin{aligned} w' &= \sum_{i=0}^{k} \varepsilon'^*_i q^i \\ &= \varepsilon'^*_k q^k + \sum_{i=0}^{k-1} \varepsilon'^*_i q^i \\ &\geq \varepsilon'^*_k q^k + \sum_{i=0}^{k-1} (-b) q^i \\ &= \varepsilon'^*_k q^k - \frac{b(q^k-1)}{q-1} \\ &= \varepsilon'^*_k q^k - \frac{(q^k-1)}{2} \\ &\geq \frac{q^k}{2} \\ &\geq q^m \\ &> q^m - 1 \end{aligned}$$

and this contradicts the fact that $w' \leq q^m - 1$.

Case the length $k < m$, relaxed reduced representation's properties in [1] show that $|\varepsilon'^*_i| \leq b$, so it is impossible to have $k < m$ with $c'(\varepsilon'^*) > 1 + bm$.

Case the length $k = m$ and $q = 3$, the relaxed cost of relaxed reduced representation is maximized when $w' = \frac{q^m+1}{2}$ which has relaxed reduced representation $\varepsilon'^* = \langle \varepsilon'^*_0, ..., \varepsilon'^*_k \rangle_q = \langle -1, ..., -1, 1 \rangle_q$. We have $c'(\varepsilon'^*) = 1 + bm$ and this is a contradiction.

Case the length $k = m$ and $q \neq 3$, if $\varepsilon'^*_k = 1$, we have $\sum_{i=0}^{k-1} \varepsilon'^*_i \geq 1 + bm$. From $|\varepsilon'^*_i| \leq b$, this is impossible. Thus, the lowest ε'^*_k we can have is 2 and we get lowest w' with relaxed cost of its relaxed reduced representation greater than $1 + bm$ is $w' = \frac{3q^m+1}{2}$ with its relaxed reduced representation $\langle \varepsilon'^*_0, ..., \varepsilon'^*_k \rangle_q = \langle -b, ..., -b, 2 \rangle_q \in R_q(w')$. You can see that $w' > q^m - 1$ and this is a contradiction. ■

Lemma 19. *For all natural number m and even base $q \geq 2$, there exists an integer $0 \leq w \leq q^m - 1$ with its relaxed reduced representation $\varepsilon^* \in R_q(w)$ such that*

$$c'(\varepsilon^*) = \begin{cases} \frac{1}{2} + \left(\frac{q-1}{2}\right)(m) & \textit{if } m \textit{ is odd} \\ 1 + \left(\frac{q-1}{2}\right)(m) & \textit{if } m \textit{ is even} \end{cases}$$

Proof. Let $b := \frac{q}{2}$ and $a := \frac{q}{2} - 1$.

Case m is odd, let $w_{odd} := q^m - aq^{m-1} - (a + bq)\frac{q^{m-1}-1}{q^2-1}$ with its representation $\varepsilon^* = \langle \varepsilon^*_0, ..., \varepsilon^*_m \rangle_q = \langle -a, -b, ..., -a, -b, -a, 1 \rangle_q \in R_q(w_{odd})$. The reduction rules in Definition 6 can be applied to show that ε^* is relaxed reduced. You can see that $0 \leq w_{odd} = \sum_{i=0}^{m} \varepsilon^*_i q^i \leq q^m - 1$ and $c'(\varepsilon^*) = 1 + a + (a + b)\left(\frac{m-1}{2}\right) = \frac{1}{2} + \left(\frac{q-1}{2}\right)(m)$.

Case m is even, let $w_{even} := q^m - \frac{q(q^m-1)}{2(q+1)}$ with its representation $\varepsilon^* = \langle \varepsilon^*_0, ..., \varepsilon^*_m \rangle_q = \langle -b, -a, ..., -b, -a, 1 \rangle_q \in R_q(w_{even})$. The reduction rules in Definition 6 can be applied to show that ε^* is relaxed reduced. You can see that $0 \leq w_{even} = \sum_{i=0}^{m} \varepsilon^*_i q^i \leq q^m - 1$ and $c'(\varepsilon^*) = 1 + (a+b)\left(\frac{m}{2}\right) = 1 + \left(\frac{q-1}{2}\right)(m)$. ■

Lemma 20. *For all natural number m and even base $q \geq 2$, $c'(\varepsilon^*)$ is at most*

$$c'(\varepsilon^*) \leq \begin{cases} \frac{1}{2} + \left(\frac{q-1}{2}\right)(m) & \textit{if } m \textit{ is odd} \\ 1 + \left(\frac{q-1}{2}\right)(m) & \textit{if } m \textit{ is even} \end{cases}$$

for relaxed reduced representation $\varepsilon^* \in R_q(w)$, $0 \leq w \leq q^m - 1$.

Proof. Let $b := \frac{q}{2}$ and $a := \frac{q}{2} - 1$.

Case m is odd, we assume that there exists an integer $0 \leq w' \leq q^m - 1$ with its relaxed reduced representation $\varepsilon'^* = \langle \varepsilon_0'^*, ..., \varepsilon_k'^* \rangle_q \in R_q(w')$ such that $c'(\varepsilon'^*) > \frac{1}{2} + \left(\frac{q-1}{2}\right)(m)$

- Case the length $k > m$, using the argument similar to Lemma 18 for the case when the length $k > m$ and some relaxed reduced representation's properties, we have $\sum_{i=0}^{k} \varepsilon_i'^* q^i > q^m - 1$. We have a contradiction.
- Case the length $k < m$, relaxed reduced representation's properties in [1] show that $|\varepsilon_i'^*| \leq b$ and there should be no two adjacent digits equal to b or $-b$, so it is impossible to have $k < m$ with $c'(\varepsilon'^*) > \frac{1}{2} + \left(\frac{q-1}{2}\right)(m)$.
- Case the length $k = m$ and $q = 2$, using the properties on adjacent digits and the argument similar to Lemma 18 for the case when the length $k = m$ and $q = 3$, we will have a contradiction.
- Case the length $k = m$ and $q \neq 2$, using the argument similar to Lemma 18 for the case when the length $k = m$ and $q \neq 3$, the lowest w' we can have with relaxed cost of its relaxed reduced representation greater than $\frac{1}{2} + \left(\frac{q-1}{2}\right)(m)$ is $w' = \frac{1}{2}\left(3q^m + 2q^{m-1} - (q+2)\left(\frac{q^{m-1}-1}{q+1}\right)\right)$ with relaxed reduced representation $\langle \varepsilon_0'^*, ..., \varepsilon_k'^* \rangle = \langle -a, -b, ..., -a, -b, -a, 2 \rangle_q \in R_q(w')$. You can see that $w' > q^m - 1$ and this is a contradiction.

Case m is even, the proof can be completed using the similar argument. ■

Finally, we conclude all lemmas in this theorem.

Theorem 21 (maximum relaxed cost of relaxed reduced representation)**.** *Given a natural number m and base $q \geq 2$, the maximum relaxed cost of relaxed reduced representation $\varepsilon^* \in R_q(w)$ for $0 \leq w \leq q^m - 1$, $c'(\varepsilon^*)$, can be described as follows.*

$$c'(\varepsilon^*) \leq \begin{cases} 1 + \left(\frac{q-1}{2}\right)(m) & \text{if } q \text{ is odd} \\ \frac{1}{2} + \left(\frac{q-1}{2}\right)(m) & \text{if } q \text{ is even and } m \text{ is odd} \\ 1 + \left(\frac{q-1}{2}\right)(m) & \text{if } q \text{ is even and } m \text{ is even} \end{cases}$$

B. *Worst Case of Computation Time Using q-Based Addition-Subtraction k-Chain Technique*

From Theorems 10, 21 and Algorithm 3, we can propose a bound of the length of q-based a/s k-chain as follows.

Theorem 22 (bound of the length of q-based addition-subtraction k-chain)**.** *Given positive integer $q \geq 2, k \geq 2$ and n, the bound of the optimal length s of q-based addition-subtraction k-chain for an integer n can be described as follows.*

$$\lceil \log_q n \rceil \leq s \leq \lceil \log_q n \rceil + \left\lceil \frac{1}{k-1}\left(\frac{q-1}{2}(\log_q n)\right)\right\rceil$$

Note that $\lceil \log_q n \rceil$ is the length of the power part and $\left\lceil \frac{1}{k-1}\left(\frac{q-1}{2}(\log_q n)\right)\right\rceil$ is the length of the addition part. This can be calculated straightforward from Theorem 10 and Theorem 21.

This theorem leads to the worst case of the computation time using q-based a/s k-chain technique (with no parallel computation). Refer to Subsection II-C, the costs from using q-based a/s k-chain of n to calculate nP are written in the following notation. We define $m := \lfloor \log_2 n \rfloor + 1$ as number of bits for n.

$$\begin{aligned} Avg_{m,(q,k)} &= \mathcal{Q} \times \frac{m}{\log_2 q} + \mathcal{A} \times c'_{avg}\left(\frac{m}{\log_2 q}\right) \\ &\quad + O(1), \\ Max_{m,(q,k)} &= \mathcal{Q} \times \frac{m}{\log_2 q} + \mathcal{A} \times c'_{max}\left(\frac{m}{\log_2 q}\right) \\ &\quad + O(1), \end{aligned}$$

where $\mathcal{Q}$ is the cost of multiplying the point by q. The constants c'_{avg} and c'_{max} are the average and maximum relaxed costs for 1 bit representation in base q which are explained in this section and Subsection II-A.

V. PARALLEL SCALAR MULTIPLICATION USING q-BASED ADDITION-SUBTRACTION k-CHAIN TECHNIQUE

In [11], k-chain is used to improve the inversion operation of an element in $GF(2^m)$. In this paper, we will use the proposed q-based a/s k-chain to improve scalar multiplication in parallel way. We first introduce the most basic technique with no parallel computation, namely 2-based a/s 2-chain technique. We then propose a 2-based a/s 4-chain parallel technique which leads to our main contribution, q-based a/s k-chain parallel technique. We also discuss an optimization for multiple operand point addition using hybrid-double multiplier [19], [20].

A. 2*-Based Addition-Subtraction* 4*-Chain Parallel Technique*

We start by introducing 2-based a/s 2-chain technique which is the most simple technique. We can calculate nP by following the way 2-based a/s 2-chain of n is constructed. We calculate point doubling in power part and point addition in the addition part of the chain. From Theorem 8, Theorem 21, Algorithms 1 and Algorithm 3, the length of the power part is at most $m := \lfloor \log_2 n \rfloor + 1$. The length of the addition part is at average $\frac{1}{3}m + O(1)$ and at most $\frac{1}{2}m + O(1)$. Thus, the average and maximum costs for 2-based a/s 2-chain technique can be written with the following notations.

$$\begin{aligned} Avg_{m,(2,2)} &= \mathcal{D} \times m + \mathcal{A} \times \frac{1}{3}m + O(1), \\ Max_{m,(2,2)} &= \mathcal{D} \times m + \mathcal{A} \times \frac{1}{2}m + O(1). \end{aligned}$$

This is the most simple technique using q-based a/s k-chain. You may notice that using NAF technique will have the same cost.

Next, we will move to the parallel technique by using parallel point addition. We start with 2-based a/s 4 chain, and then generalize to q-based a/s k chain in the next subsection. The idea is that, to calculate $P_1 + P_2 + P_3 + P_4$, parallel point addition will firstly do $P' = P_1 + P_2$ and $P'' = P_3 + P_4$ simultaneously with 2 computing cores. Then the final answer is $P' + P''$. Thus, this costs only $2\mathcal{A}$ instead of $3\mathcal{A}$. We define 4-operand parallel point addition $\mathcal{A}_{||}$ which equals to $2\mathcal{A}$.

Remark: Given elliptic points P_i and positive integer x, x-operand point addition is the calculation of $P_1+P_2+\cdots+P_x$.

To use 2-based a/s 4-chain parallel technique to calculate nP, we do point doubling in power part and then do 4-operand parallel point addition in addition part. The length of the power part is at most $m := \lfloor \log_2 n \rfloor + 1$. By Theorems 8 and 22, the length of the addition part is $\frac{1}{3}(\frac{1}{3})m + O(1)$ at average and, by Theorems 21 and 22, at most $\frac{1}{3}(\frac{1}{2})m + O(1)$. Thus, the average and maximum costs for 2-based a/s 4-chain parallel point addition technique with 2 computing cores can be written with the following notations.

$$\begin{aligned} Avg_{m,(2,4,2)} &= \mathcal{D} \times m + \mathcal{A}_{\|} \times \frac{1}{9}m + O(1) \\ &= \mathcal{D} \times m + \mathcal{A} \times \frac{2}{9}m + O(1), \end{aligned}$$

$$\begin{aligned} Max_{m,(2,4,2)} &= \mathcal{D} \times m + \mathcal{A}_{\|} \times \frac{1}{6}m + O(1) \\ &= \mathcal{D} \times m + \mathcal{A} \times \frac{1}{3}m + O(1). \end{aligned}$$

Another easy way to calculate the cost is to multiply a factor of $\frac{2}{3}$ from the parallel point addition to the addition part of 2-based a/s 2-chain.

B. Generalize to q-Based Addition-Subtraction k-Chain Technique with c Computing Cores

If you have enough computing resources, this technique can be generalized by using q-based a/s k-chain with c computing cores (can execute c point additions at the same time). Start with k operands (points), because it is a k-chain, we assume that number of the operands are reduced by half each round (with 2-operand addition), thus we have $\lceil \log_2 k \rceil$ rounds. The number of point additions in each round is the number of the operands remaining in each round divides by 2 times the number of computing cores, rounding up. Similar to the last subsection, the average and maximum costs for the generalized technique can be written as follows.

$$\begin{aligned} Avg_{m,(\text{odd } q,k,c)} &= \mathcal{Q} \times \frac{m}{\log_2 q} + \mathcal{A} \times \left(1 - 2^{-\lceil \log_2 k \rceil}\right) \times \\ &\quad \left(\frac{k(q-1)(q+1)}{4cq(k-1)}\right) \frac{m}{\log_2 q} + O(1) \\ Avg_{m,(\text{even } q,k,c)} &= \mathcal{Q} \times \frac{m}{\log_2 q} + \mathcal{A} \times \left(1 - 2^{-\lceil \log_2 k \rceil}\right) \times \\ &\quad \left(\frac{k(q-1)(q+2)}{4c(q+1)(k-1)}\right) \frac{m}{\log_2 q} + O(1) \\ Max_{m,(q,k,c)} &= \mathcal{Q} \times \frac{m}{\log_2 q} + \mathcal{A} \times \left(1 - 2^{-\lceil \log_2 k \rceil}\right) \times \\ &\quad \left(\frac{k(q-1)}{2c(k-1)}\right) \frac{m}{\log_2 q} + O(1) \end{aligned}$$

Another easy way to approximate the cost is to multiply a factor of $\frac{1}{c}$ from the parallel computing cores to the addition part.

Remark: To get the minimum cost, we found that we should use $q = 2$ (2-based) and have the highest c as possible. (The choices for k have only small effect. We will discuss on this topic in the next section.) Moreover, using $q = 2$ is also efficient in $GF(2^m)$ because the doubling operation is fast. The value of q can be chosen depending on the finite field used.

C. Optimizing Multiple Operand Point Addition for Twisted Edwards Curves

In this subsection, we discuss an optimization technique for q-based a/s k-chain. Although we can use any q for q-based a/s k-chain technique, the last subsection, Subsection V-B, show that $q = 2$ will lead to the minimum cost and we will use this value through this subsection as an example on how to apply the optimization. (Other values of q are also possible for this technique.)

Firstly, we give some explanations about the finite field elements multiplication. The 2-operand finite field elements multiplication is an operation that can give a multiplication result of two field elements, while the 3-operand finite field elements multiplication is an operation that can give a multiplication result of three field elements. For example, when $\lambda_1, \lambda_2, \lambda_3$ are elements of a field, the 2-operand finite field elements multiplication can give a value of $\lambda_1\lambda_2$, while 3-operand finite field elements multiplication can give a value of $\lambda_1\lambda_2\lambda_3$.

In [19], [20], a circuit for the 3-operand finite field elements multiplication called hybrid-double multiplier is given. The calculation time of the circuit is almost equal to that of the fastest 2-operand finite field elements multiplication.

Remark: The finite field elements multiplication is a part of the elliptic point addition. They are not the same operations.

We will focus on twisted Edwards curves [18], $ax^2 + y^2 = 1 + dx^2y^2$, because they are the currently fastest curves [21]. Cost for 2-operand point addition (an operation that calculate $P_1 + P_2$ when P_1 and P_2 are points on an elliptic curve) using hybrid-double multiplier is $\mathcal{A} = 9\mathcal{M}$. We define $\mathcal{A}^*$ as the cost for 3-operand point addition (an operation that calculate $P_1 + P_2 + P_3$ when P_1, P_2, and P_3 are points on an elliptic curve). We can rearrange some part of the point addition equations from [18] and have $17\mathcal{M}$ as the cost for $\mathcal{A}^*$. The rearranged equations for $\mathcal{A}^*$ are explained as follows.

Let X_i, Y_i and Z_i be finite field elements. Given elliptic points $P_1(X_1, Y_1, Z_1)$, $P_2(X_2, Y_2, Z_2)$ and $P_3(X_3, Y_3, Z_3)$ in projective coordinate where the x-y coordinate $(x_i, y_i) = (X_i/Z_i, Y_i/Z_i)$. We use most equations from [18] because they are the currently fastest formulas recorded in [21]. First, we calculate X_{12} and Y_{12} from P_1 and P_2.

$$\begin{aligned} \lambda_1 &= Z_1Z_2 \\ \lambda_2 &= (\lambda_1)^2 \\ \lambda_3 &= X_1X_2 \\ \lambda_4 &= Y_1Y_2 \\ \lambda_5 &= d\lambda_3\lambda_4 \\ \lambda_6 &= \lambda_2 - \lambda_5 \\ \lambda_7 &= \lambda_2 + \lambda_5 \end{aligned}$$

$$
\begin{aligned}
X_{12} &= \lambda_1\lambda_6((X_1+Y_1)(X_2+Y_2)-\lambda_3-\lambda_4) \\
Y_{12} &= \lambda_1\lambda_7(\lambda_4-a\lambda_3)
\end{aligned}
$$

The calculation cost is now $8\mathcal{M}$ using hybrid-double multiplier. Notice that we do not calculate Z_{12} here. (Calculate $Z_{12}=\lambda_6\lambda_7$ will finish the 2-operand point addition with the cost of $9\mathcal{M}$.) We continue to calculate $P_4(X_4,Y_4,Z_4)=P_1+P_2+P_3$ as follows and you can see that the total cost is $17\mathcal{M}$.

$$
\begin{aligned}
\lambda_8 &= \lambda_6\lambda_7 Z_3 \\
\lambda_9 &= (\lambda_8)^2 \\
\lambda_{10} &= X_{12}X_3 \\
\lambda_{11} &= Y_{12}Y_3 \\
\lambda_{12} &= d\lambda_{10}\lambda_{11} \\
\lambda_{13} &= \lambda_9-\lambda_{12} \\
\lambda_{14} &= \lambda_9+\lambda_{12} \\
X_4 &= \lambda_8\lambda_{13}((X_{12}+X_3)(Y_{12}+Y_3)-\lambda_{10}-\lambda_{11}) \\
Y_4 &= \lambda_8\lambda_{14}(\lambda_{11}-a\lambda_{10}) \\
Z_4 &= \lambda_{13}\lambda_{14}
\end{aligned}
$$

It is important to note that with the hybrid-double multiplier, we have $17\mathcal{M}=\mathcal{A}^* < 2\mathcal{A}=18\mathcal{M}$. The lower cost comes from the λ_8 equation. To calculate nP from 2-based a/s k-chain, we do point doubling in the power part and then do 3-operand point addition in the addition part. In this way, the average and maximum costs for 2-based a/s k-chain technique with c cores are

$$
\begin{aligned}
Avg_{m,\text{optimize}(2,k,c)} &= \mathcal{D}\times m+\mathcal{A}\times\left(1-3^{-\lceil\log_3 k\rceil}\right) \\
&\quad\times\left(\frac{17k}{54c(k-1)}\right)m+O(1) \\
Max_{m,\text{optimize}(2,k,c)} &= \mathcal{D}\times m+\mathcal{A}\times\left(1-3^{-\lceil\log_3 k\rceil}\right) \\
&\quad\times\left(\frac{17k}{36c(k-1)}\right)m+O(1)
\end{aligned}
$$

Another easy way to calculate the cost is to multiply a factor of $\frac{17}{18}$ from the 3-operand point addition to the addition part.

We can generalize this optimization to x-operand point addition for any positive integer $x\geq 3$. Calculating the summation of x elliptic points with hybrid-double multiplier but no multiple operand point addition will cost $9(x-1)\mathcal{M}$. For each operand higher than 2, if we use the hybrid-double multiplier, we can reduce the finite field multiplication by one. This is because we do not have to calculate the intermediate value of Z_i. Refer to the rearrange equations earlier, you can see that we can calculate $\lambda_8=\lambda_6\lambda_7Z_3$ with only one 3-operand field elements multiplication instead of $Z_{12}=\lambda_6\lambda_7$ and $\lambda_8=Z_{12}Z_3$ separately. You can apply this 3-operand field element multiplication to every point operand added. This means that, using x-operand point addition will reduce $(x-2)\mathcal{M}$ from the cost. This will give a factor of $\frac{9(x-1)-(x-2)}{9(x-1)}=\frac{8x-7}{9x-9}$ to the addition part.

Recall that the number of points we want to add using c cores is k (from the property of q-based a/s k-chain). We know that the computation time is optimized when each core has almost the same number of points to add. Because of that, each core has the point to calculate the summation at most $\left\lceil\frac{k}{c}\right\rceil$. Having $x>\left\lceil\frac{k}{c}\right\rceil$ will not give any more improvements. Thus, the value of x that minimize our cost is $\left\lceil\frac{k}{c}\right\rceil$.

If we can rearrange the equations in other ways and reduce more field element multiplications, we can similarly apply the technique to lower the calculation cost. However, we found during the course of this work that the further reduction is very difficult.

D. Side Channel Attack Prevention

In previous techniques including double-and-add, NAF, w-NAF, addition chain and addition-subtraction chain, we can know the sequence of addition and doubling steps by monitoring the power (power attack) [12] or time used (timing attack) [13]. These kinds of attacks are called side-channel attack. Some techniques, including Montgomery ladder [14], deploy dummy operations or use uniform operations to encounter this, however, the calculation costs become worse. In q-based a/s k-chain techniques, all calculations begin with some numbers of multiplying and then follow with some additions. The n in nP cannot be distinguished using these kind of sequences. Thus, we do not need any dummy operations. This is how we resist power attack.

For timing attack in q-based a/s k-chain, one can measure the time used in the addition part and tell how many times the addition are executed. Some n may have short addition part, thus, it is easy to guess this integer n. We can add some dummy operations and make all executions equal to the maximum computation time. Even though the calculation cost becomes worse, we have proved that the maximum computation time is still efficient. In this way, we can prevent timing attack and also power attack.

VI. Comparisons and Parameter Settings

A. Comparisons with Techniques without Windows

Defining $m:=\lfloor\log_2 n\rfloor+1$, from Sections V, the costs Avg_m and Max_m of all techniques using base 2 representations are in the form

$$\mathcal{D}\times m+\mathcal{A}\times\text{coeff. } m+O(1).$$

Table I summarizes the coefficient values of $\mathcal{A}m$ for average and worst case. w-NAF and wr-NAF are not included in the table because their calculation costs are affected from the precomputation costs.

Note that the multiple operand point addition optimization is now specific to twisted Edwards curves due to the rearranged equations. The other techniques can be generally applied to any curves.

According to Table I, among the proposed technique, choosing 2-based a/s k-chain with large number of cores, c, and a multiple operand point addition optimization will lead to the minimum cost.

TABLE I
COEFFICIENT VALUE OF $\mathcal{A}m$ FOR EACH TECHNIQUE

Technique used	**Average coefficient**	**Maximum coefficient**
Double-and-add	$\frac{1}{2} = 0.5$	1
2-based a/s 2-chain / NAF [2]	$\frac{1}{3} \approx 0.333...$	$\frac{1}{2} = 0.5$
2-based a/s 4-chain (2 cores)	$\frac{2}{9} \approx 0.222...$	$\frac{1}{3} \approx 0.333...$
2-based a/s k-chain (c cores)	$\left(1 - 2^{-\lceil \log_2 k \rceil}\right)\left(\frac{k}{3c(k-1)}\right) \approx \frac{1}{3c} \approx \frac{0.333...}{c}$	$\left(1 - 2^{-\lceil \log_2 k \rceil}\right)\left(\frac{k}{2c(k-1)}\right) \approx \frac{1}{2c} \approx \frac{0.5}{c}$
2-based a/s k-chain (c cores) with 3-operand addition and hybrid-double multiplier	$\left(1 - 3^{-\lceil \log_3 k \rceil}\right)\left(\frac{17k}{54c(k-1)}\right) \approx \frac{17}{54c} \approx \frac{0.315...}{c}$	$\left(1 - 3^{-\lceil \log_3 k \rceil}\right)\left(\frac{17k}{36c(k-1)}\right) \approx \frac{17}{36c} \approx \frac{0.472...}{c}$
2-based a/s k-chain (c cores) with x-operand addition and hybrid-double multiplier	$\left(1 - x^{-\lceil \log_x k \rceil}\right)\left(\frac{k(8x-7)}{3c(k-1)(9x-9)}\right) \approx \frac{8}{27c} \approx \frac{0.296...}{c}$	$\left(1 - x^{-\lceil \log_x k \rceil}\right)\left(\frac{k(8x-7)}{2c(k-1)(9x-9)}\right) \approx \frac{4}{9c} \approx \frac{0.444...}{c}$

B. Comparisons with w-NAF and Montgomery Ladder

We first precisely compare the maximum cost between w-NAF representation and our 2-based a/s k-chain parallel technique. Because of the pre-computation cost in w-NAF, we have to specify the number of bits used to represent an integer. We choose 128 and 256 bit integers for this purpose [22]. The maximum cost should be considered in this case due to the aims of side-channel attack. For w-NAF (wr-NAF with $r = 2$), [4] suggests the best w to be 5 and we will use these values in this subsection. According to Subsections II-C and V-B, the maximum cost for 128 bits would be

$$\begin{aligned} Max_{128,\text{5-NAF}} &= \mathcal{D} \times 128 + \mathcal{A} \times \left(\frac{1}{5}(128) + 7\right) \\ &= 128\ \mathcal{D} + \frac{163}{5}\mathcal{A}, \\ Max_{128,(2,k,c)} &= \mathcal{D} \times 128 + \mathcal{A} \times \left(1 - 2^{-\lceil \log_2 k \rceil}\right) \\ &\quad \times \left(\frac{k}{2c(k-1)}\right) 128 + O(1) \end{aligned}$$

In this case, using 2-based a/s k-chain technique with only 2 computing cores will have slightly smaller cost. You can improve the result by using more than 2 computing cores. Next, the maximum cost for 256 bits would be

$$\begin{aligned} Max_{256,\text{5-NAF}} &= \mathcal{D} \times 256 + \mathcal{A} \times \left(\frac{1}{5}(256) + 7\right) \\ &= 256\ \mathcal{D} + \frac{291}{5}\mathcal{A}, \\ Max_{256,(2,k,c)} &= \mathcal{D} \times 256 + \mathcal{A} \times \left(1 - 2^{-\lceil \log_2 k \rceil}\right) \\ &\quad \times \left(\frac{k}{2c(k-1)}\right) 256 + O(1) \end{aligned}$$

In 256-bit case, using 2-based a/s k chain technique with 3 computing cores or more will have smaller cost.

Approximately, for large m, our chain technique with 3 computing cores will have up to $\frac{1/5 - 4/27}{1/5} = 25.92\%$ less additions. but if using 2 computing cores, our technique can have up to $\frac{4/18 - 1/5}{1/5} = 11.11\%$ more additions.

Similarly, in base 3 representation and $GF(3^m)$, [4] suggests $w = 3$ and $r = 3$ for wr-NAF technique. Our 3-based a/s k-chain technique can overcome the current cost using only 3 computing cores with 3-operand addition optimization and hybrid-double multiplier. We show the approximated costs as follows and use $\mathcal{T}$ as the cost for tripling.

$$\begin{aligned} Max_{m,\text{(3,3)-NAF}} &= \mathcal{T} \times \frac{m}{\log_2 3} + \mathcal{A} \times \left(\frac{1}{3}\right) \frac{m}{\log_2 3} \\ &\quad + O(1) \\ Max_{m,\text{opt}(3,k,c)} &= \mathcal{T} \times \frac{m}{\log_2 3} + \mathcal{A} \times \left(\frac{17}{18c}\right) \frac{m}{\log_2 3} \\ &\quad + O(1) \end{aligned}$$

For Montgomery ladder [14], the technique uses 1 addition for each digit in the binary representation. Using the number of bits with the same notation $m = \lfloor \log_2 n \rfloor + 1$, the maximum cost for calculating nP using this technique is

$$Max_{m,Mont} = \mathcal{A} \times m$$

Because the cost of addition $\mathcal{A} = 11\mathcal{M}$ is more than doubling $\mathcal{D} = 7\mathcal{M}$ (with no hybrid-double multiplier), our 2-based a/s k-chain techniques can perform better with 2 computing cores and above. If the power part is already pre-computed, our chain technique can perform better in any case.

C. Choice of k in q-Based Addition-Subtraction k-Chain

Although, choices for k do not have much effects on the cost, you can see that choosing large k for q-based a/s k-chain can lower the calculation cost owing to the cost equation, the rounding up and the optimization technique. However, using smaller k also has some benefits.

In elliptic curve cryptography, we know P long before knowing n. The power part can be calculated before n is given. When n is given, it will go through Algorithm 1 to generate its relaxed reduced representation. After the algorithm produces some least significant digits, if we choose small k, we can calculate the next element of the chain immediately, not having to wait for the algorithm to finish. The smaller k we choose, the earlier we can start the calculation.

Smaller k also has another benefit. Some applications need the results of nP for the same P and some different n. Using smaller k, some common digits between the different n can be calculated only once. See the example below.

Example 23. *To calculate $45P$ and $77P$ from 2-based a/s k-chain, we have relaxed reduced representation for $45 = \langle 1, 0, -1, 0, -1, 0, 1 \rangle_2$ and $77 = \langle 1, 0, -1, 0, 1, 0, 1 \rangle_2$. From Algorithm 3, we can choose x_i (each digit in relaxed reduced representation) in order to have t for $45 = (1, -3, -5, 0)$ and t for $77 = (1, -3, 5, 0)$. If we choose $k = 3$ (2-based a/s 3-chain), we have $[1, 2, 4, 8, 16, 32, 64, 61, 45]_2$ for 45 and $[1, 2, 4, 8, 16, 32, 64, 61, 77]_2$ for 77. Notice that we can calculate the common $61P$ once. We can reduce more addition if n is large. In this example, the technique cannot be used if we choose $k \geq 4$.*

From our experiments, we random 100,000 pairs of 128-bit integers and generate their base 2 relaxed reduced representations. At average, there are 3 common digits between each pair. Changing to base 3 can increase this value to 17. This means if we choose the appropriate value for k, at average, we can reduce 3 and 17 additions, respectively. We may use this technique and consider more than 2 integers at the same time. A method for selecting which digit to calculate, its efficiency and method to find the optimized k are left for further studies.

VII. Conclusions and Future Works

Parallel techniques have been used widely for efficient computations, but not many parallel techniques have been proposed to improve scalar multiplication with side-channel attack prevention. Our paper propose a parallel scalar multiplication technique using q-based a/s k-chain. The proposed technique uses the computing cores efficiently as we can see that using c computing cores can reduce the addition costs to the factor of $\frac{1}{c}$. This parallel techniques can perform better than the current w-NAF technique by using only 3 cores. We also have some optimization of multiple operand point addition which can reduce the addition costs up to the factor of $\frac{8}{9}$. At the end, we show that our parallel q-based a/s k-chain technique can resist side-channel attack efficiently. Some elliptic curve multiple operand addition equations rearrangements, method for selecting digits and the optimized value for k are left for further studies.

Acknowledgment

This research was mainly done while Kittiphop Phalakarn and Kittiphon Phalakarn did an internship at The University of Tokyo. The authors would like to thank The University of Tokyo, Assis. Prof. Athasit Surarerks and Prof. Hiroshi Imai for facilitating the internship.

References

[1] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.
[2] W. Bosma, "Signed bits and fast exponentiation," *Journal de théorie des nombres de Bordeaux*, vol. 13, no. 1, pp. 27–41, 2001.
[3] K. Okeya and T. Takagi, "The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks," in *Cryptographers Track at the RSA Conference*, pp. 328–343, Springer, 2003.
[4] T. Takagi, D. J. Reis, S.-M. Yen, and B.-C. Wu, "Radix-r non-adjacent form and its application to pairing-based cryptosystem," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 89, no. 1, pp. 115–123, 2006.
[5] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, 1997.
[6] H. Volger, "Some results on addition/subtraction chains," *Information Processing Letters*, vol. 20, no. 3, pp. 155–160, 1985.
[7] N. Meloni, "Fast and secure elliptic curve scalar multiplication over prime fields using special addition chains," *IACR Cryptology ePrint Archive*, vol. 2006-216, 2006.
[8] R. R. Goundar, K. Shiota, and M. Toyonaga, "New strategy for doubling-free short addition-subtraction chain," *International Journal of Applied Mathematics*, vol. 2, no. 3, pp. 438–445, 2007.
[9] H. M. Bahig, "On a generalization of addition chains: Addition–multiplication chains," *Discrete Mathematics*, vol. 308, no. 4, pp. 611–616, 2008.
[10] M. Nöcker, "Some remarks on parallel exponentiation," in *International Symposium on Symbolic and Algebraic Computation*, pp. 250–257, ACM, 2000.
[11] K. Järvinen, V. Dimitrov, and R. Azarderakhsh, "A generalization of addition chains and fast inversions in binary fields," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2421–2432, 2015.
[12] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*, pp. 388–397, Springer, 1999.
[13] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Annual International Cryptology Conference*, pp. 104–113, Springer, 1996.
[14] M. Joye and S.-M. Yen, "The Montgomery powering ladder," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 291–302, Springer, 2002.
[15] N. Méloni and M. A. Hasan, "Random digit representation of integers," in *IEEE Symposium on Computer Arithmetic (in press)*, 2016.
[16] B. Möller, "Securing elliptic curve point multiplication against side-channel attacks," in *International Conference on Information Security*, pp. 324–334, Springer, 2001.
[17] C. Heuberger and H. Prodinger, "On minimal expansions in redundant number systems: Algorithms and quantitative analysis," *Computing*, vol. 66, no. 4, pp. 377–393, 2001.
[18] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, "Twisted Edwards curves," in *International Conference on Cryptology in Africa (AfricaCrypt '08)*, pp. 389–405, Springer, 2008.
[19] R. Azarderakhsh and A. Reyhani-Masoleh, "Low-complexity multiplier architectures for single and hybrid-double multiplications in Gaussian normal bases," *IEEE Transactions on Computers*, vol. 62, no. 4, pp. 744–757, 2013.
[20] R. Azarderakhsh and A. Reyhani-Masoleh, "Parallel and high-speed computations of elliptic curve cryptography using hybrid-double multipliers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1668–1677, 2015.
[21] "Explicit-formulas database." http://hyperelliptic.org/EFD/. Accessed July 21, 2016.
[22] A. Jivsov, "Elliptic curve cryptography (ECC) in OpenPGP," 2012.