New Addition Operation and Its Application for Scalar Multiplication on Hessian Curves over Prime Fields *

Leibo Li, Mingqiang Wang, Zhanjiang Xia
Key Laboratory of Cryptologic Technology and Information Security,
Ministry of Education, Shandong University, 250100 Jinan, China

Abstract

In this paper, we present a new addition operation on Hessian curves with low cost. It can be applied to resist the side channel attacks for scalar multiplication, and also can be used to compute precomputation points for window-based scalar multiplication on Hessian curves over prime fields. We propose two new precomputation schemes that are shown to achieve the lowest cost among all known methods. By using the fractional wNAF and fractional wmbNAF, if $n = 192$ bits and $1I \approx 30M$, scheme 1 can save up to $31M$, scheme 2 can save up to $28M$ with $w \geq 6$, where I, M represent the inversion and the multiplication, respectively.

Keywords: Elliptic curve cryptosystem, scalar multiplication, Hessian curves

1 Introduction

Since Elliptic Curve Cryptography (ECC) was introduced by Koblitz and Miller in 1985, this cryptosystem has attracted increasing attention due to its shorter key size requirement in comparison to other public-key cryptosystem. Scalar multiplication ($[n]P$) is the central operation of elliptic curve cryptosystem, where n is the scalar and P is a point on an elliptic curve over F_p. Important methods for scalar multiplication are window non-adjacent form (wNAF) and fractional window non-adjacent form (Frac-wNAF) [9], they are based on the binary expansion of number. This means that the binary expansion can be directly translated to computation using the simplest elementary ECC point operation, namely point doubling and addition. The Hamming weight of the

*This work is supported by national 973. Grant No.2007CB807903, and nature science of Shandong province Grant No Y2008G23.

binary expansion of the scalar can be reduced by precomputation, and thus reduce the number of addition operation, furthermore, reduce the cost of scalar multiplication. In 2007, Longa and Miri [7] have proposed the multibase non-adjacent form (mbNAF) and fractional window multibase non-adjacent form (Frac-wmbNAF), they used a very efficient representation of integer using multiple base. Efficiently of these methods depend on the cost of addition operation and doubling operation. The addition operation has the form of $dP^{(i)}+Q$, where d is small integer (≥ 2), Q is in the set of $\{3P, 5P, ..., (2L+1)P\}$. Thus, it is a critical task to reduce the computing cost of these operations, to further speed up the executive time of scalar multiplication.

In 2006, Meloni [5] introduced a special addition operation in Jacobian coordinate on Elliptic curves. In 2008, Patrick Longa [2] proposed a new methodology to perform the precomputation in Frac-wNAF using this addition operation, which used only one field inversion to convert all points in precomputation set $\{3P, 5P, ..., (2k+1)P\}$ to affine coordinates, and use the mixed coordinates addition to achieve the addition operation, then reduce the cost of scalar multiplication.

In this paper, we introduce a new technique to perform addition operation on Hessian curves with low cost, which add two points with same z coordinate. For any fixed n, we can get the addition sequence v using the method in [5], where

$$v = (v_1, ..., v_s),\ v_1 = 1,\ v_s = n,$$

and satisfying $v_i = v_{i-1} + v_j$ where $1 \leq j < i-1$. By combining our method and the addition chains, we have an iterative computation of the form $P_i = P_{i-1} + P_j$, where $P_1 = P,\ P_s = nP$, and every addition can be computed with the identical z coordinate. Thus, scalar multiplication can be performed efficiently with only point addition (no doubling), that can resist the side channel attacks. Our method can also be used to drive faster the window-based scalar multiplication. Given that precomputation points in window-based algorithm follow the form d_iP, where d_i is an odd integer ≥ 3. In the second part of this paper, we propose two schemes to perform the precomputation. In scheme 1, we use the iterative computation of the form $d_iP = 2P + ... + 2P + 2P + P$, which perform the point addition with new operation, and all points are converted to affine representation using the method in [2], the conversion need only one point inversion. Thus, we can use mixed coordinates addition to perform scalar multiplication. In comparison to the method in projective coordinates, if $I/M \approx 30$, $n = 192$ bits, scheme 1 could save $47M$ by using Frac-wNAF, and $36M$ by using Frac-wmbNAF. In scheme 2, we use the new addition operation to perform precomputation in projective coordinates. In comparison to the previous method, scheme 2 could save up to $28M$ with $w \geq 6$ by using Frac-wNAF and Frac-wmbNAF.

This paper is organized as follows. Section 2 introduces the basic of Hessian curves and special addition operation by Meloni. Section 3 introduces a new addition operation on Hessian Curves and the method to perform the scalar multiplication which can resist the side channel attacks. Section 4 proposes two schemes for precomputation and the comparison to traditional methods. Section 5 analyse the cost of our schemes and the traditional methods for scalar multiplication by using Frac-wNAF and Frac-wmbNAF. Section 6 states the conclusion.

2 Preliminaries

2.1 Basic of Hessian Curves

Hessian curves in cryptology are explained by Joye and Quisquater [3], and Smart [4]. An elliptic curve over K in Hessian form is defined by

$$x^3 + y^3 + 1 = 3dxy,$$

where $d \in K$ with $d^3 \neq 1$. Hessian-form elliptic curve is birationally equivalent to the Weierstrass-form elliptic curve

$$v^2 = u^3 - 27d(d^3 + 8)u + 54(d^6 - 20d^3 - 8).$$

A typical point $(x,\ y)$ on the Hessian curve corresponds to the point $(u,\ v)$ on the Weierstrass curve defined by $u = p - 9d^2$ and $v = 3p(y - x)$, where $p = 12(d^3-1)/(d+x+y)$. The identity element is the point at infinity. The negative of a point $(x,\ y)$ is $(y,\ x)$. The addition formula is $(x_3,\ y_3) = (x_1,\ y_1)+(x_2,\ y_2)$, where

$$x_3 = \frac{y_1^2 x_2 - y_2^2 x_1}{x_2 y_2 - x_1 y_1}, \qquad y_3 = \frac{x_1^2 y_2 - x_2^2 y_1}{x_2 y_2 - x_1 y_1}.$$

The doubling formula is $(x_3,\ y_3) = 2(x_1,\ y_1)$, where

$$x_3 = \frac{y_1(1 - x_1^3)}{x_1^3 - y_1^3}, \qquad y_3 = \frac{x_1(y_1^3 - 1)}{x_1^3 - y_1^3}.$$

On projective coordinates, each point is represented by the triplet $(X : Y : Z)$ which satisfies the projective curve $X^3+Y^3+Z^3 = 3dXYZ$ and corresponds to the affine point $(X/Z : Y/Z)$ with $Z \neq 0$. The identity element is represented by $(1 : -1 : 0)$. The negative of $(X : Y : Z)$ is $(Y : X : Z)$. For all nonzero $\lambda \in K$, $(X : Y : Z) = (\lambda X : \lambda Y : \lambda Z)$. The point addition formula in projective coordinates is given by $(X_3 : Y_3 : Z_3) = (X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)$, where

$$\begin{aligned} X_3 &= Y_1^2 X_2 Z_2 - Y_2^2 X_1 Z_1, \\ Y_3 &= X_1^2 Y_2 Z_2 - X_2^2 Y_1 Z_1, \\ Z_3 &= Z_1^2 X_2 Y_2 - Z_2^2 X_1 Y_1. \end{aligned}$$

The point doubling formula in projective coordinates is given by $(X_3 : Y_3 : Z_3) = 2(X_1 : Y_1 : Z_1)$, where

$$\begin{aligned} X_3 &= Y_1(Z_1^3 - X_1^3), \\ Y_3 &= X_1(Y_1^3 - Z_1^3), \\ Z_3 &= Z_1(X_1^3 - Y_1^3). \end{aligned}$$

The point addition costs $12M$ with the projective coordinates, $10M$ with the affine-projective mixed coordinates addition, point doubling costs $7M + 1S$ and tripling costs $8M + 6S$.

2.2 Special Addition Operation by Meloni

In 2006, Nicolas Meloni [5] proposed a new addition operation with low cost in Jacobian coordinate on Elliptic curves, which is described in the follow.

Let $P = (X_1 : Y_1 : Z)$ and $Q = (X_2 : Y_2 : Z)$ be two points with the same z coordinate in jacobian coordinates on the elliptic curve E, two points addition $P + Q = (X_3 : Y_3 : Z_3)$ can be obtained as follows:

$$\begin{aligned} X_3 &= (Y_2 - Y_1)^2 - (X_2 - X_1)^3 - 2X_1(X_2 - X_1)^2, \\ Y_3 &= (Y_2 - Y_1)(X_1(X_2 - X_1)^2 - X_3) - Y_1(X_2 - X_1)^3, \\ Z_3 &= Z(X_2 - X_1). \end{aligned} \tag{1}$$

This special addition operation costs $5M + 2S$. Meloni applied this addition operation to compute the scalar multiplication by using the addition chains. An addition chains which compute an integer n is given by two sequences v and w such that

$$\begin{aligned} v &= (v_1, ..., v_s),\ v_1 = 1,\ v_2 = 2,\ v_3 = 3,\ v_s = k, \\ w &= (w_4, ..., w_s) \in \{0,1\}^{s-2},\ where \\ v_{i+1} &= v_i + v_j \begin{cases} w_{i+1} = 0 & : \quad j = i - 1 \\ w_{i+1} = 1 & : \quad 1 \le j < i - 1. \end{cases} \end{aligned} \tag{2}$$

Let $P_{i+1} = P_i + P_{i-1}$ and $P_i = P_{i-1} + P_j$, where P_{i-1} and P_j have the same z coordinate, we can use the formulae (1) to perform $P_i = P_{i-1} + P_j$. Because the coordinate of P_{i-1} can be written as

$$\begin{aligned} (X_{i-1} : Y_{i-1} : Z_{i-1}) &\equiv (X_{i-1}(X_j - X_{i-1})^2 : Y_{i-1}(X_j - X_{i-1})^3 \\ &\quad : Z_{i-1}(X_j - X_{i-1})) \\ &= (X_{i-1}(X_j - X_{i-1})^2 : Y_{i-1}(X_j - X_{i-1})^3 : Z_i). \end{aligned}$$

Thus, it is possible to perform $P_{i+1} = P_i + P_{i-1}$ use the formulae (1) without any extra addition. In total, it costs $(5s-6)M + (2s+2)S$ for scalar multiplication,

where s is the length of the addition chains. In 2008, Patrick Longa [2] use this addition operation to compute the precomputation of window-based scalar multiplication. He use this operation to convert all percomputation points to affine coordinates with only one point inversion. In comparison to previous methods, his schemes deliver the lowest cost for scalar multiplication by using Frac-wNAF.

3 New Addition Operation on Hessian curves

This section introduces the new addition operation on Hessian curves. Let $P = (X_1 : Y_1 : Z)$ and $Q = (X_2 : Y_2 : Z)$ be two points with the same z coordinate in projective coordinates on the Hessian curves. The addition $P + Q = (X_3 : Y_3 : Z_3)$ can be obtained as follows:

$$\begin{aligned} X_3 &= Z(Y_1^2 X_2 - Y_2^2 X_1), \\ Y_3 &= Z(X_1^2 Y_2 - X_2^2 Y_1), \\ Z_3 &= Z^2(X_2 Y_2 - X_1 Y_1). \end{aligned}$$

$(X_3 : Y_3 : Z_3)$ can alteratively be written as

$$\begin{aligned} X_3 &= (Y_2 - Y_1)(X_2 Y_2 - (Y_1 + Y_2)(X_1 + X_2)) + Y_1(X_2 Y_2 - X_1 Y_1), \\ Y_3 &= (X_2 - X_1)(X_2 Y_2 - (Y_1 + Y_2)(X_1 + X_2)) + X_1(X_2 Y_2 - X_1 Y_1), \\ Z_3 &= Z(X_2 Y_2 - X_1 Y_1). \end{aligned} \tag{3}$$

Using formulae (3), two points with same z coordinate addition only costs $8M$.

Using the addition sequences (2) in section 2.1, we can perform the scalar multiplication with low cost. Let $P_1 = (X_1 : Y_1 : Z_1)$ be a point in Hessian curves and n is a positive integer, the point doubling

$$P_2 = 2P_1 = (Y_1(Z_1^3 - X_1^3) : X_1(Y_1^3 - Z_1^3) : Z_1(X_1^3 - Y_1^3)) = (X_2 : Y_2 : Z_2).$$

Then the point P_1 can be represented as

$$P_1 = (X_1 : Y_1 : Z_1) \equiv (X_1(X_1^3 - Y_1^3) : Y_1(X_1^3 - Y_1^3) : Z_1(X_1^3 - Y_1^3)).$$

Thus P_1 and P_2 have the same z coordinate, we can use the formulae (3) to compute $P_1 + P_2 = P_3$, it costs $7M + 1S$ for $2P$, $2M$ for precomputation and $8M$ for addition.

Assuming w_4 in the formulae (2) is 1, and $P_3 + P_1$ will be computed according to the addition chains. Because the coordinates of P_1 can be represented as

$$(X_1(X_2 Y_2 - X_1 Y_1) : Y_1(X_2 Y_2 - X_1 Y_1) : Z_1(X_2 Y_2 - X_1 Y_1)),$$

it was computed in the course of $P_1 + P_2$, so we can use the formulae (3) to perform the addition $P_1 + P_3$, it does not require any precomputation. In total, scalar multiplication require $(s-2)$ times additions with same z coordinate and one doubling, all of these operations cost $(8s-7)M + 1S$, where s represents the length of addition chains.

Side channel attacks have been discovered by Kocher in 1996. The attacker can deduce secret information by the method to analyse the amount of time required to perform secret operation. This weakness mainly depends on the fact that during the computation of scalar multiplication, addition operation is more expensive than doubling operation. Thus a side-channel analysis allows to deduce what kind of operation is computed, and further to guess the bit of the exponent. The efficient measure to avoid this attack is using the dummy operations, that makes the group operation look identical during the computing process. By combining the new addition operation in this paper and addition chains, we can perform the scalar multiplication using only point additions with low cost, so it is possible to perform scalar multiplication efficiently and resist the side channel attacks efficaciously.

4 Application in Precomputation

By using windows-based method to perform scalar multiplication, it is required to precompute the following set:

$$\{3P, 5P..., (2L+1)P\} = \{3P, 5P..., mP\}, \tag{4}$$

where L is number of precomputation points. In this section, we will propose two schemes for precomputation on Hessian curves using the new addition operation, then analyse the cost in comparison to traditional methods for Frac-$wNAF$.

4.1 Method for Precomputation

The precomputation set can be proposed as follows:

$$d_i P = ... + 2P + 2P + 2P + P. \tag{5}$$

Scheme 1 : Let point $P = (x_1 : y_1)$ is assumed originally to be in affine coordinates.

Step 1. We can compute doubling $2P$ in affine coordinates and yield result in projective coordinates as follows:

$$\begin{aligned} X_2 &= y_1(1 - x_1^3) = y_1 - y_1^4 - \alpha, \\ Y_2 &= x_1(y_1^3 - 1) = x_1^4 - x_1 - \beta, \\ Z_2 &= x_1^3 - y_1^3 = (x_1 - y_1)(x_1^2 + x_1 y_1 + y_1^2), \end{aligned} \tag{6}$$

where $\alpha = y_1(x_1^3 - y_1^3)$, $\beta = x_1(x_1^3 - y_1^3)$. This formula is easily derived from the doubling formula in section 2 and only costs $3M + 5S$. Then, by fixing $\lambda = x_1^3 - y_1^3$, we can assume the following equivalent point to P,

$$P^{(1)} = (X_1^{(1)} : Y_1^{(1)} : Z_1^{(1)}) = (x_1(x_1^3 - y_1^3) : y_1(x_1^3 - y_1^3) : (x_1^3 - y_1^3)),$$

which does not require any extra costs since its coordinates have already been computed in formulae (6).

Step 2. We will compute precomputation points d_iP in projective coordinates, which be computed using formulae (6) as follows.

1^{st},
$3P = 2P + P^{(1)} = (X_2 : Y_2 : Z_2) + (X_1^{(1)} : Y_1^{(1)} : Z_1^{(1)}) = (X_3 : Y_3 : Z_3)$,
$X_3 = (Y_1^{(1)} - Y_2)(X_1^{(1)}Y_1^{(1)} - (Y_2 + Y_1^{(1)})(X_2 + X_1^{(1)})) + Y_2(X_1^{(1)}Y_1^{(1)} - X_2Y_2)$,
$Y_3 = (X_1^{(1)} - X_2)(X_1^{(1)}Y_1^{(1)} - (Y_2 + Y_1^{(1)})(X_2 + X_1^{(1)})) + X_2(X_1^{(1)}Y_1^{(1)} - X_2Y_2)$,
$Z_3 = Z_2(X_1^{(1)}Y_1^{(1)} - X_2Y_2)$.

2^{st},
$(2P)^{(1)} = (X_2^{(1)} : Y_2^{(1)} : Z_2^{(1)}) = (X_2(X_1^{(1)}Y_1^{(1)} - X_2Y_2) : Y_2(X_1^{(1)}Y_1^{(1)} - X_2Y_2) :$
$Z_2(X_1^{(1)}Y_1^{(1)} - X_2Y_2)) \equiv (X_2 : Y_2 : Z_2)$,
$5P = 3P + (2P)^{(1)} = (X_3 : Y_3 : Z_3) + (X_2^{(1)} : Y_2^{(1)} : Z_2^{(1)}) = (X_4 : Y_4 : Z_4)$,
$X_4 = (Y_3 - Y_2^{(1)})(X_3Y_3 - (Y_2^{(1)} + Y_3)(X_2^{(1)} + X_3)) + Y_2^{(1)}(X_3Y_3 - X_2^{(1)}Y_2^{(1)})$,
$Y_4 = (X_3 - X_2^{(1)})(X_3Y_3 - (Y_2^{(1)} + Y_3)(X_2^{(1)} + X_3)) + X_2^{(1)}(X_3Y_3 - X_2^{(1)}Y_2^{(1)})$,
$Z_4 = Z_2^{(1)}(X_3Y_3 - X_2^{(1)}Y_2^{(1)})$, $A_4 = (X_3Y_3 - X_2^{(1)}Y_2^{(1)})$.
...
...
$((m-1)/2)^{st}$,
$(2P)^{((m-3)/2)} = (X_2^{((m-3)/2)} : Y_2^{((m-3)/2)} : Z_2^{((m-3)/2)}) = (X_2^{((m-5)/2)}(X_{(m-3)/2}$
$Y_{(m-3)/2} - X_2^{((m-5)/2)}Y_2^{((m-5)/2)}) : Y_2^{((m-5)/2)}(X_{(m-3)/2}Y_{(m-3)/2} - X_2^{((m-5)/2)}$
$Y_2^{((m-5)/2)}) : Z_2^{((m-5)/2)}(X_{(m-3)/2}Y_{(m-3)/2} - X_2^{((m-5)/2)}Y_2^{((m-5)/2)}))$,
$mP = (2P)^{((m-3)/2)} + (m-2)P = (X_2^{((m-3)/2)} : Y_2^{((m-3)/2)} : Z_2^{((m-3)/2)}) +$
$(X_{(m+1)/2} : Y_{(m+1)/2} : Z_{(m+1)/2}) = (X_{(m+3)/2} : Y_{(m+3)/2} : Z_{(m+3)/2})$,
$X_{(m+3)/2} = (Y_{(m+1)/2} - Y_2^{((m-3)/2)})(X_{(m+1)/2}Y_{(m+1)/2} - (Y_2^{((m-3)/2)} + Y_{(m+1)/2})$
$(X_2^{((m-3)/2)} + X_{(m+1)/2})) + Y_2^{((m-3)/2)}(X_{(m+1)/2}Y_{(m+1)/2} - X_2^{((m-3)/2)}Y_2^{((m-3)/2)})$,
$Y_{(m+3)/2} = (X_{(m+1)/2} - X_2^{((m-3)/2)})(X_{(m+1)/2}Y_{(m+1)/2} - (Y_2^{((m-3)/2)} + Y_{(m+1)/2})$
$(X_2^{((m-3)/2)} + X_{(m+1)/2})) + X_2^{((m-3)/2)}(X_{(m+1)/2}Y_{(m+1)/2} - X_2^{((m-3)/2)}Y_2^{((m-3)/2)})$,
$Z_{(m+3)/2} = Z_2^{((m-3)/2)}(X_{(m+1)/2}Y_{(m+1)/2} - X_2^{((m-3)/2)}Y_2^{((m-3)/2)})$,

$A_{(m+3)/2} = (X_{(m+1)/2}Y_{(m+1)/2} - X_2^{((m-3)/2)}Y_2^{((m-3)/2)})$.

Value A_i, for $i = 4$ to $(m+3)/2$, are stored and used in the following for converting all points to affine coordinates.

Step 3. Converting points $(X_i : Y_i : Z_i)$, for i from 3 to $(m+3)/2$ to affine coordinates. Thus, we can use the mixed coordinates addition to perform scaler multiplication. The conversion can be achieved by means of $(X_i/Z_i : Y_i/Z_i : 1)$. We first compute the inversion $r = Z_{(m+3)/2}^{-1}$, and then recover every point as follows.

$mP:\ x_{(m+3)/2} = rX_{(m+3)/2},\ y_{(m+3)/2} = rY_{(m+3)/2},$
$(m-2)P:\ r = rA_{(m+3)/2},\ x_{(m+1)/2} = rX_{(m+1)/2},\ y_{(m+1)/2} = rY_{(m+1)/2},$
...
...
$3P:\ r = rA_4,\ x_3 = rX_3,\ y_3 = rY_3.$

Theorem 1 *In total, this methodology cost*

$$Cost_{scheme\ 1} = 1I + (11L + 2)M + 5S \tag{7}$$

to compute $[3]P, [5]P.....[m]P$*, where* $L = (m-1)/2$ *represents the number of precomputation points. Furthermore, this methodology requires* $3L + 2$ *registers for temporary calculations and points storage, for* $L = 1$*, the requirement is fixed at 5 registers.*

Proof. Step 1 costs $3M + 5S$ for computing $2P$ and $P^{(1)}$, step 2 costs $(8L)M$ for computing the points d_iP in projective coordinates, step 3 costs $(3L-1)M + 1I$ for converting all points to affine coordinates. Then the sum of all steps cost $1I + (11L + 2)M + 5S$.

Regarding memory requirements. The point P needs two registers $T_1^{(1)}$ and $T_2^{(1)}$ to store x_1 and y_1. Doubling computation $2P$ need $T_3^{(1)}$, $T_4^{(1)}$ and $T_5^{(1)}$ to store X_2, Y_2 and Z_2, $T_1^{(1)}$ and $T_2^{(1)}$ are reused to store α and β. For the first addition $2P + P^{(1)}$, $T_3^{(1)}$, $T_4^{(1)}$ and $T_5^{(1)}$ are reused to store X_3, Y_3 and Z_3, $T_1^{(1)}$ and $T_2^{(1)}$ are reused to store $X_2^{(1)}$ and $Y_2^{(1)}$. Thus, if $L = 1$, we need 5 registers to preform the addition. For the second addition, we need $T_1^{(2)}$, $T_2^{(2)}$ and $T_3^{(2)}$ to store X_4, Y_4 and A_4. $T_1^{(1)}$, $T_2^{(1)}$ and $T_5^{(1)}$ are reused to store $X_2^{(2)}$, $Y_2^{(2)}$ and Z_4. Similarly, we need 3 extra registers to store the new points for each addition when $L \geq 2$. In conclusion, this method requires $3L + 2$ registers in the end of step 4.

There are different efficient schemes to compute precomputation points in the literature. The simplest approaches suggest performing computation in $\mathcal{A}$ or

$\mathcal{P}$ using the chains $P \to 3P \to 5P \to ... \to mP$, which requires one doubling and $L = (m-1)/2$ additions, where $\mathcal{A}$ and $\mathcal{P}$ represent affine coordinates and projective coordinates, respectively. For the precomputation in $\mathcal{A}$, which requires $3M + 3S + 1I$ for the doubling $2P$ and $6M + 1I$ for each addition. For the precomputation in $\mathcal{P}$, we consider that doubling $2P$ is computed as $2\mathcal{A} \to \mathcal{P}$ with cost of $3M + 3S$, the first addition $P + 2P$ is computed using a mixed addition as $\mathcal{A} + \mathcal{P} \to \mathcal{P}$ with cost of $10M$, the following $(L-1)$ additions have the form of $\mathcal{P} + \mathcal{P} \to \mathcal{P}$ $(12M)$. In total,

$$Cost_{\mathcal{A}} = (L+1)I + (6L+3)M + 3S, \quad (8)$$

$$Cost_{\mathcal{P}} = (12L+1)M + 3S. \quad (9)$$

Storing the points requires $2L + 2$ registers for affine coordinates and $3L + 3$ registers for projective coordinates.

Scheme 2 : The doubling $2P$ is computed as the formulae (6) with the cost of $3M + 5S$, the first addition $P + 2P$ is computed using new addition operation with same z coordinate, and the following $(L-1)$ additions is also computed using the new addition operation too. In this case, all of the precomputation points are in projective coordinates.

Theorem 2 *In total, this methodology cost*

$$Cost_{scheme\ 2} = (8L+5)M + 3S \quad (10)$$

to compute $[3]P, [5]P.....[m]P$*, where* $L = (m-1)/2$ *represents the number of points. Furthermore, this methodology also requires* $3L + 2$ *registers for temporary calculations and storing precomputation points, for* $L = 1$*,the requirement is fixed at 5 registers.*

Similar proof as theorem 1.

4.2 Performance Comparison

The advantage of a method depends on the ratio of inversions and multiplication I/M and the ratio of squaring and multiplication S/M, where I, S and M represent the inversion, squaring and multiplication operation, respectively. In this analysis, the S/M ratio is set to $1S \approx 0.8M$. Then we estimate the I/M ratio of our method and the straightforward precomputation in $\mathcal{A}$.

Table 1. I/M ranges for scheme 1 and the straightforward precomputation in $\mathcal{A}$

points	2	3	4	5	6	7	8
Schema 1	≥ 5.3	≥ 5.2	≥ 5.15	≥ 5.12	≥ 5.1	≥ 5.09	≥ 5.07
Affine	≤ 5.3	≤ 5.2	≤ 5.15	≤ 5.12	≤ 5.1	≤ 5.09	≤ 5.07

Table 1 shows the I/M values for scheme 1 and the straightforward precomputation in $\mathcal{A}$ for a given number of precomputation points. As it can be seen, scheme 1 is superior to the straightforward precomputation in $\mathcal{A}$ when inversion is more then 5.3 times of the cost of multiplication, and it is usually expected that $I/M \geq 30$.

Furthermore, by using Frac-wNAF for scalar multiplication, we compare the cost of scheme 1, scheme 2 and $\mathcal{P}$-based method, whose cost is given by the formula (8), (11) and (10), respectively. When precomputation is preformed in $\mathcal{P}$, we would use the form $\mathcal{P} + \mathcal{P} \rightarrow \mathcal{P}$ to compute additions $(12M)$, and $2\mathcal{P} \rightarrow \mathcal{P}$ to compute every doubling $(7M + 1S)$. Then the scalar multiplication costs including precomputation are as follows,

$$[n\mathcal{D}(19M + 1S) + n(1 - \mathcal{D})(7M + 1S)] + [(12L + 1)M + 3S], \tag{11}$$

where $\mathcal{D}$ represents the average non-zero density in Frac-wNAF, and

$$\mathcal{D} = [\lfloor log_2 m \rfloor + \frac{(m+1)}{2^{\lfloor log_2 m \rfloor}} + 1]^{-1}. \tag{12}$$

If we use the scheme 2 to compute precomputation, costs of scalar multiplication are as follows,

$$[n\mathcal{D}(19M + 1S) + n(1 - \mathcal{D})(7M + 1S)] + [(8L + 5)M + 3S]. \tag{13}$$

For scheme 1, we use $\mathcal{P} + \mathcal{A} \rightarrow \mathcal{P}$ to perform mixed additions $(10M)$, and $2\mathcal{P} \rightarrow \mathcal{P}$ to every doubling $(7M + 1S)$. Then the scalar multiplication costs including precomputation are as follows,

$$[n\mathcal{D}(17M + 1S) + n(1 - \mathcal{D})(7M + 1S)] + [1I + (11L + 2)M + 5S]. \tag{14}$$

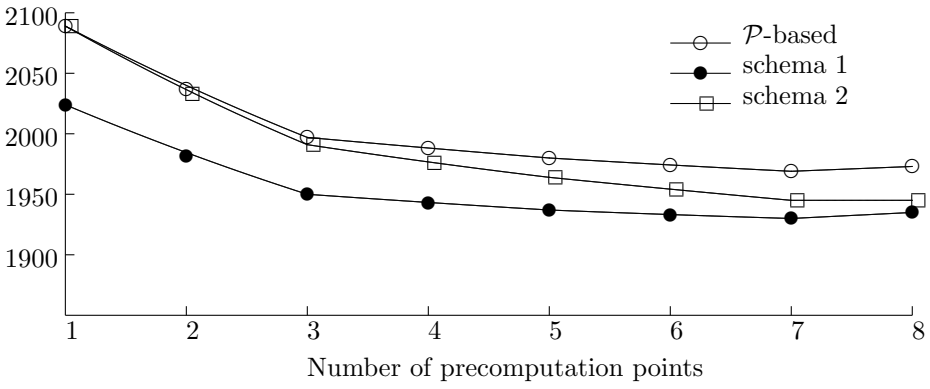

Fig.1 cost performance of scheme 1, scheme 2 and $\mathcal{P}$-based method

Fig.1 represents costs performance of scheme 1 (14), scheme 2 (13) and $\mathcal{P}$-based

method (12) to perform scalar multiplication including precomputation by using Frac-wNAF, assuming $1I \approx 30M, 1S \approx 0.8M, n = 192$bits. As we can see, the total cost for scalar multiplication is minimal if $L = 7$ or $m = 15$. In comparison with the $\mathcal{P}$-based method, scheme 1 can save up to $39M$, it can also save $47M$ or 2.35% with $w = 4$, scheme 2 can save up to $28M$ with $w \geq 6$.

5 Application in Frac-wmbNAF

Multibase NAF (mbNAF) was proposed by Longa in 2007 [12], using this method we can find a multibase chain which has balance in the number of addition and all the other point operations. It has the form

$$k = \sum_{i=1}^{m} s_i \prod_{j=1}^{J} a_j^{c_i(j)},$$

where a_j is prime integer from a set of base $\{a_1 \cdots a_J\}$, m is the length of the expansion. If a larger window size is allowed, we could reduce further the average number of nonzero terms in scalar multiplication, this technique is named Frac-wmbNAF.

If we use the base $\{2, 3\}$, window size is w and L is the number of precomputation points, then the average densities of addition, doubling and triplings is

$$\begin{aligned} \mathcal{D}_a &= \frac{2^w}{8(L+1) - 3(u+v) + 2^{w-2}(4w-1)}, \\ \mathcal{D}_2 &= \frac{8(L+1) + 2^w(w-1)}{8(L+1) - 3(u+v) + 2^{w-2}(4w-1)}, \\ \mathcal{D}_3 &= \frac{3(2^{w-2} - (u+v))}{8(L+1) - 3(u+v) + 2^{w-2}(4w-1)}, \end{aligned} \tag{15}$$

where $u = \lfloor (L+2)/3 \rfloor$ and $v = \lfloor (2^{w-2} - L)/3 \rfloor$. Then, we will analysis the cost of scalar multiplication using Frac-wmbNAF on Hessian curves. If precomputation points coordinates are in $\mathcal{P}$, then the total cost of scalar multiplication is

$$\begin{aligned} Cost_{\mathcal{P}-based}^{Frac-wmbNAF} &= (\mathcal{D}_a 12M + \mathcal{D}_2(7M + 1S) + \mathcal{D}_3(8M + 6S))digits \\ &\quad + Cost_{\mathcal{P}}, \end{aligned} \tag{16}$$

$$\begin{aligned} Cost_{scheme\ 2}^{Frac-wmbNAF} &= (\mathcal{D}_a 12M + \mathcal{D}_2(7M + 1S) + \mathcal{D}_3(8M + 6S))digits \\ &\quad + Cost_{scheme\ 2}, \end{aligned} \tag{17}$$

where $Cost_{\mathcal{P}}$ is given by formula (10) and $Cost_{schema\ 2}$ is given by (11), $digits$ represents the total number of digits in the expansion,

$$digits = \frac{(n-1)\lg 2}{\mathcal{D}_2 \lg 2 + \mathcal{D}_3 \lg 3}.$$

If the precomputation points coordinates are in $\mathcal{A}$, then the total cost of scalar multiplication is,

$$\begin{aligned} Cost_{scheme\ 1}^{Frac-wmbNAF} &= (\mathcal{D}_a 10M + \mathcal{D}_2(7M + 1S) + \mathcal{D}_3(8M + 6S))digits \\ &\quad + Cost_{scheme\ 1}. \end{aligned} \tag{18}$$

where $Cost_{scheme\ 1}$ is given by formula (8). Table 2 shows costs of Frac-wNAF with scheme 1 (14), scheme 2 (13), $\mathcal{P}$-based method (12), and Frac-wmbNAF with scheme 1 (18), scheme 2 (17) and $\mathcal{P}$-based method (16) on Hessian curves over prime field. Assuming $n = 192$ bits, $1S \approx 0.8M$, L represents the number of precomputation point.

Table 2, costs of Frac-wNAF and Frac-wmbNAF with various precomputation methods

L	1	2	3	4
$Cost_{\mathcal{P}-based}^{Frac-wNAF}$	$2089M$	$2037M$	$1997M$	$1988M$
$Cost_{scheme\ 1}^{Frac-wNAF}$	$1994M + 1I$	$1952M + 1I$	$1920M + 1I$	$1913M + 1I$
$Cost_{scheme\ 2}^{Frac-wNAF}$	$2089M$	$2033M$	$1989M$	$1976M$
$Cost_{\mathcal{P}-based}^{Frac-wmbNAF}$	$2021M$	$1920M$	$1926M$	$1931M$
$Cost_{scheme\ 1}^{Frac-wmbNAF}$	$1939M + 1I$	$1850M + 1I$	$1860M + 1I$	$1865M + 1I$
$Cost_{scheme\ 2}^{Frac-wmbNAF}$	$2021M$	$1916M$	$1917M$	$1918M$
L	5	6	7	8
$Cost_{\mathcal{P}-based}^{Frac-wNAF}$	$1980M$	$1974M$	$1969M$	$1973M$
$Cost_{scheme\ 1}^{Frac-wNAF}$	$1907M + 1I$	$1903M + 1I$	$1900M + 1I$	$1905M + 1I$
$Cost_{scheme\ 2}^{Frac-wNAF}$	$1964M$	$1954M$	$1945M$	$1945M$
$Cost_{\mathcal{P}-based}^{Frac-wmbNAF}$	$1927M$	$1916M$	$1922M$	$1914M$
$Cost_{scheme\ 1}^{Frac-wmbNAF}$	$1872M + 1I$	$1855M + 1I$	$1861M + 1I$	$1854M + 1I$
$Cost_{scheme\ 2}^{Frac-wmbNAF}$	$1911M$	$1896M$	$1898M$	$1885M$

We have computed the cost of Frac-wNAF and Frac-wmbNAF with different window size for 192 bits scalar multiplication. In the case of Frac-wmbNAF, we use the base set of $\{2, 3\}$. For each of method, we consider three cases: precomputation points are left in projective coordinates with the ordinary computation (refer to the case $Cost_{\mathcal{P}-based}^{Frac-wNAF}$ and $Cost_{\mathcal{P}-based}^{Frac-wmbNAF}$), precomputation points are left in projective coordinates with the new addition operation (refer to the case $Cost_{scheme\ 2}^{Frac-wNAF}$ and $Cost_{scheme\ 2}^{Frac-wmbNAF}$), precomputation points are converted to affine coordinates using one point inversion (refer to the case $Cost_{scheme\ 1}^{Frac-wNAF}$ and $Cost_{scheme\ 1}^{Frac-wmbNAF}$). As it can be seen, when $w \geq 6$, using scheme 2 for precomputation in $\mathcal{P}$ could reduce up to $28M$ comparing to the ordinary method. If the I/M was small, we could convert the precomputation points to $\mathcal{A}$ using scheme 1. When $I/M \approx 30$, the cost could be saved up to $31M$. When $w = 4$, scheme 1 could save $47M$ by using Frac-wNAF, and $36M$ by using Frac-wmbNAF. We also give the lowest performance of scalar multiplication on Hessian curves, when $w = 6$ and 6 points for precomputation with

scheme 1 by using Frac-wmbNAF. Furthermore, it is important to note that the improvement would be even more significant in implementations where a hardware multiplier executes both squaring and multiplication (i.e., $1S \approx 1M$).

6 conclusion

In this paper, we have described the new addition operation on Hessian curves, it only costs $8M$ in addition computation with same z coordinate, and we also remarked that scalar multiplication could avoid the side-channel attacks by using the new addition operation and addition chains. In the second part of this paper, we presented two efficient precomputation method using the new addition operation, scheme 1 need only one inversion to convert all of precomputation point to affine coordinates, scheme 2 reduce the cost of precomputation in projective coordinates. Then we described that proposed methods are more efficient than the general method for scalar multiplication over prime fields. We also showed that our methods offer the lowest costs.

References

[1] Daniel J. Bernstein and Tanja Lange: Explicit-formulas database(2007). http://www.hyperelliptic.org/EFD.

[2] Partrick Longa and Ali Miri: New composite opereations and precomputation scheme for Elliptic curves cryptosystems over prime fields. In: *PKC 2008*, LNCS, 4939,pp. 229-247. International Association for Cryptologic Research (2008).

[3] Marc Joye and Jean-Jacques Quisquater: Hessian Elliptic Curves and Side-Channel Attacks. In: *CHES 2001*. LNCS 2162, pp. 402C410, 2001. Springer-Verlag Berlin Heidelberg (2001).

[4] N.P. Smart: The Hessian Form of an Elliptic Curve. In: *CHES 2001*, LNCS 2162, pp. 118C125, 2001. Springer-Verlag Berlin Heidelberg (2001).

[5] Nicoal Meloni: Fast and Secure Elliptic Curve Scalar Multiplication Over Prime Fields Using Special Addition Chains. Cryptology ePrint Archive, Report 2006/216 (2006).

[6] H.Cohen and G.Frey, editors: Handbook of Elliptic and Hyperelliptic Curve Eryptography. CRC press (2005).

[7] Patrick Longa and Catherine Gebotys: Fast Multibase Methods and Other Several Optimizations for Elliptic Curve Scalar Multiplication. In: *PKC 2009*, lNCS, 5443,pp,443-462. International Association for Cryptologic Research (2009).

[8] Erik Dahmen, Katsuyuki Okeya and Daniel Schepers: Affine Precomputation with Sole Inversion in Elliptic Curve Cryptography. In: *ACISP 2007*,LNCS 4586, pp. 245C258, 2007. Springer-Verlag Berlin Heidelberg (2007).

[9] Bodo Moller: Improved Techniques for Fast Exponentiation. In: *ICISC 2002*,LNCS, vol.2587,pp.298-312. Springer Heidelberg (2003).

[10] Vassil Dimitrov, Laurent Imbert and Pradeep Kumar Mishra: Efficient and Secure Elliptic Curve Point Multiplication Using Double-Base Chains. In: *ASIACRYPT 2005*,LNCS 3788, pp. 59C78, 2005. International Association for Cryptologic Research (2005).

[11] Patrick Longa and Ali Miri: New Multibase Non-Adjacent Form Scalar Multiplication and its Application to Elliptic Curve Cryptosystem. (submitted 2007).

[12] Patrick Longa and Catherine Gebotys: Fast multibase methods and Other several Optimuzations for Elliptic Curve Scalar Multiplication. In: *PKC2009* LNCS 5443, pp. 443C462, 2009. International Association for Cryptologic Research (2009).