An Improved Algorithm for $uP + vQ$ Using $\mathrm{JSF}_3^{1\star}$

BaiJie Kuang, YueFei Zhu, and YaJuan Zhang

Network Engineering Department Information Engineering University, Zhengzhou, 450002, P.R.China
{kbj123,zyf0136,springzyj}@sina.com

Abstract. Techniques for fast exponentiation (multiplication) in various groups have been extensively studied for use in cryptographic primitives. Specifically the joint expression of two exponents (multipliers) plays an important role in the performances of the algorithms used. The crucial optimization relies in general on minimizing the joint Hamming weight of the exponents (multipliers).

J.A.Solinas suggested an optimal signed binary representation for pairs of integers, which is called a Joint Sparse Form (JSF) [25]. JSF is at most one bit longer than the binary expansion of the larger of the two integers, and the average joint Hamming density among Joint Sparse Form representations is 1/2.

This paper extends the Joint Sparse Form by using a window method, namely, presents a new representation for pairs of integers, which is called Width-3 Joint Sparse Form (JSF_3), and proves that the representation is at most one bit longer than the binary expansion of the larger of the two integers and its average joint Hamming density is 37.1% via the method of stochastic process. So, Computing the form of $uP + vQ$ by using JSF_3 is almost 8.6% faster than that by using JSF.

1 Introduction

Known to all, the design of the Public Key Cryptosystem mostly depends on the particular algebra construction. The basic public-key operation in a finite field F_q is to compute g^a for a given element $g \in F_q$ and a positive integer a. This is typically accomplished by the the binary method [6], based on the binary expansion of a. The method requires $\sim l/2$ general multiplications and $\sim l$ squarings (on average). $(l = \lceil \log_2 q \rceil)$.

More generally, it is needed to evaluate expressions of the form $g^a h^b$. In particular, most common digital signatures (RSA,ECDSA) are verified by evaluating an expression of the above. This is typically accomplished by the Straus' Methods [5,6,2]. The method requires $\sim l$ general multiplications and $\sim l$ squarings (on average). After then, numerous methods of speeding up scalar multiplication have been discussed in the literature; for a survey, see [8].

⋆ This work was supported by NSF(No.90204015), Found 973(No.G1999035804), and Elitist Youth Foundation of HeNan Province(No.021201400) in China.

M. Jakobsson, M. Yung, J. Zhou (Eds.): ACNS 2004, LNCS 3089, pp. 467–478, 2004.

While on general Elliptic Curve $E(F_q), P = (x, y) \in E(F_q)$, then $-P = (x, -y)$. Thus point subtraction is as efficient as addition. This property motivates using a signed binary expansion (allowing coefficients 0 and ± 1). A particularly useful signed digit representation is the non–adjacent form (NAF) [3,24]. By using a window method, one processes some other signed digit representation, called the width–w nonadjacent form (NAF_w) [1,3,8,24]. (when w=2, NAF_w is equivalent to NAF). There is a simply and efficient algorithm for presenting NAF_w of any integer. When computing kP, the method requires $\sim l/(w+1)$ general point additions and $\sim l$ doubles.

Furthermore, many Elliptic Curve Cryptosystems require the computation of the form $uP+vQ$, where P, Q are points on an elliptic curve and u, v are integers, such as verification schemes of ECDSA [10]. In the following, we will call this form multi scalar multiplications. So the efficiency of implementation depends mostly on the efficiency of evaluation of multi scalar multiplications. Thus, fast multi scalar multiplications is essential for Elliptic Curve Cryptosystems. There are lots of research papers on the problem of speeding up $uP + vQ$ in the recent years [1,2,3,7,8,12,16,21,25,26].

Computing the form $uP + vQ$, J.A.Solinas [25] suggested an optimal signed binary representation for pairs of integers, called Joint Sparse Form (JSF). There is an algorithm to product JSF for pairs of integers. And it is at most one bit longer than the binary expansion of the larger of the two integers, and the average joint Hamming density among Joint Sparse Form representations is 1/2.

In [25], Solinas remarks that a generalization would allow coefficients other than ± 1. Avanzi [1] presents an analogue of JSF with windows, whose average joint Hamming density is 3/8.

This paper also extends the JSF by using some other signed digit representation of integers and presents the concept of the form representation of integers, and brings forward Width–3 Joint Sparse Form (JSF_3). At last it also proves the average joint Hamming density (AJHD) is 37.1% via the method of stochastic process. So, this improvement can speed the computation of the form $uP + vQ$ by up to 8.6%, compared to compute that by using JSF. Computing $uP + vQ$ by using JSF_3 wins that by using the other previous forms.

The rest of the paper is organized as follows. In Section 2, we briefly review elliptic curves and give some preparation knowledge on the representation of integers. Section 3 first gives the definition of JSF_3 for pairs of positive integers u_1, u_2, then proves its existence, i.e. presents an algorithm for producing it, and last shows AJHD of that is 37.1% via stochastic process. Section 4 gives the application of the technique and discusses avenues for further work.

2 Preparation Knowledge

2.1 Elliptic Curves

Up to a birational equivalence, an *elliptic curve* over a field **K** is a plane nonsingular cubic curve with a **K**–rational point [22]. Elliptic curves are often expressed

in terms of Weierstraß equations:

$$E/\mathbf{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

where $a_1, \cdots, a_6 \in \mathbf{K}$. If characteristic Char $(\mathbf{K}) \neq 2, 3$, the equation may be simplified to $y^2 = x^3 + a_4x + a_6$, and if Char($\mathbf{K}$)=2 the equation (for a non-supersingular curve) may be simplified to $y^2 + xy = x^3 + a_2x^2 + a_6$.

Together with an extra point $\mathcal{O}$, the points on an elliptic curve form an Abelian group. We use the additive notation. The scalar multiplication is the form:

$$kP = \underbrace{P + P + \cdots + P}_{k\ times}.$$

And multi scalar multiplications is the form $uP + vQ$. The crucial optimization relies in general on minimizing the joint Hamming weight of the two multipliers.

2.2 Expansion of Integer

A given nonegative integer n has a common binary expansion

$$n = (a_{l-1}, \cdots, a_1, a_0) = \sum_{i=0}^{l-1} a_i 2^i, \quad a_i = 0, 1.$$

and integer n has another binary expansion

$$n = (b_{t-1}, \cdots, b_1, b_0) = \sum_{i=0}^{t-1} b_i 2^i,\ b_i \in \{0, \pm 1, \pm 3, \cdots, \pm(2^{w-1} - 1)\}, (w > 0).$$

We call it the width–w generalized (binary) expansion form of n (GF_w). Obviously, there are many such expansions. We say that GF_w is *reduced* if the expansion has the property that the product of any w consecutive terms is nonegative. More, the reduced GF_w is width–w non adjacent form (NAF_w) if the expansion has the property that there is at most a nonzero term of any w consecutive terms. We know, every integer has unique NAF_w [3,24]. There is also a simple and efficient algorithm for computing the NAF_w of a given integer. The NAF_w of a positive integer is at most one bit longer than its binary expansion, and the NAF_w has the minimal Hamming weight among GF_ws of n. Namely, The average Hamming density among NAF_w is $1/(w + 1)$ [3,24].

Let n be a positive integer, the notation "$n\ mods8$" denotes that the modular reduction 8 is to return the smallest residue in absolute value. Correspondingly for Width–3 generalized expansion of $n, n = (a_{l-1}, \cdots, a_1, a_0)$, obviously, $a_0 = 0$ if n is an even number; and if n is an odd number, then

$$a_0 \in \{\ n\ mods8,\ (n + 4)\ mods8,\ -(n\ mods8),\ -((n + 4)\ mods8)\ \}.$$

So, we may call a_0

Fetching-Original-Value of n (FOV(n)), if $a_0 = n\ mods8$;
Fetching-Anti-Value of n (FAV(n)), if $a_0 = (n + 4)\ mods8$;
Fetching-Sign-Value of n (FSV(n)), if $a_0 = -(n\ mods8)$;
Fetching-Number-Value of n (FNV(n)), if $a_0 = -((n + 4)\ mods8)$.

Example 1: For an integer n=13, $n = (a_{l-1}, \cdots, a_0)$.
If $a_0 = n\ mods8$, namely, $a_0 = -3$, then n=(1,0,0,0,-3), a_0=FOV(n).
If $a_0 = (n + 4)\ mods8$, namely, $a_0 = 1$, then n=(1,0,-1,0,1), a_0=FAV(n).

If $a_0 = -(n \ mods 8)$, namely, $a_0 = 3$, then n=(1,0,1,3), a_0=FSV(n).
If $a_0 = -((n+4) \ mods 8)$, namely, $a_0 = -1$, then n=(1,0,3,-1), a_0=FNV(n).

From the example above, we can draw that:

Lemma 1. For an integer n, $n = (a_{l-1}, \cdots, a_0)$, we can have that:
If a_0=FOV(n), then $a_1 = 0, a_2 = 0$.
If a_0=FAV(n), then $a_1 = 0, a_2 \neq 0$.
If a_0=FSV(n) or a_0=FNV(n), then $a_1 \neq 0$.

Definition 1. (joint Hamming weight (JHW)) [26] Let n_0, n_1 be two l–bit elements of N. Considering the $2 \times l$ array whose rows are the signed expansions of the elements, we say that the joint Hamming weight (JHW) of n_0 and n_1 with the expansions form is the number of nonzero columns of the array and denote JHW of n_0, n_1 by JHW(n_0, n_1) (JHW for short). The average joint Hamming density (AJHD) is the ratio of JHW(n_0, n_1) to its length, where n_0, n_1 run over l–bit elements N.

2.3 JSF for Pairs of Integers

Computing the form $uP + vQ$, J.A.Solinas [25] suggested an optimal signed binary representation for pairs of integers, called Joint Sparse Form (JSF). The expansion takes on the following properties:

(JSF–1.) Among three consecutive columns at least one is a double zero.
(JSF–2.) It is never the case that $u_{i,j+1} \cdot u_{i,j} \neq -1$.
(JSF–3.) If $u_{i,j+1} \cdot u_{i,j} \neq 0$, then $u_{1-i,j} = 0, u_{1-i,j+1} \neq 0$.

There is an algorithm to product JSF for arbitrary pairs of integers. JSF is at most one bit longer than the binary expansion of the larger of the two integers, and the average joint Hamming density among Joint Sparse Form representations is 1/2.

3 JSF_3 for Pairs of Integers

We call the joint width–3 generalized expansions for integers n_0, n_1 the width 3–joint generalized expansion form of n_0, n_1 (JGF_3). More, we call it the reduced width–3 generalized expansion form of n_0, n_1 (JRF_3) if both are reduced. Analogically, call it the joint NAF_3 (JNF_3). It isn't difficult to see that JHW of JNF_3 is quite smaller, but it is not the smallest among all JGF_3s. Thereinafter, we give the expansion that is quite small among JGF_3s, whose AJHD is 37.1%, while that of JNF_3 is 43.8%.

Definition 2. The joint width–3 generalized expansion for integers n_0, n_1,

$$n_0 = (u_{0,m-1}, \cdots, u_{0,1}, u_{0,0}).$$
$$n_1 = (u_{1,m-1}, \cdots, u_{1,1}, u_{1,0}).$$

is called Width–3 Joint Sparse Form ($\mathrm{JSF}_3(n_0, n_1)$), shortly noted by JSF_3, if the expansion satisfies the following conditions:

($\mathbf{JSF}_3$–1.) Of any three consecutive columns, at least one is zero,and of any five consecutive columns, at least two are zeros.
($\mathbf{JSF}_3$–2.) For every row, the product of adjacent terms is not -1.
($\mathbf{JSF}_3$–3.) If $\exists i \in \{0,1\}$ satisfies $u_{i,j} \neq 0,\ u_{i,j+1} \neq 0$,then, $u_{1-i,j+1} \neq 0$, and $u_{1-i,j} = 0$.
($\mathbf{JSF}_3$–4.) If $\exists i \in \{0,1\}$ satisfies $u_{i,j} \neq 0,\ u_{i,j+2} \neq 0$, then $u_{1-i,j+2} \neq 0$.

Example 2. For two integers $n_0 = 2365$ and $n_1 = 2921$, we have the JSF_3 shown below:

$$n_0 = (\ 1,\ 0,\ 0,\ 0,\ 1,\ 3,\ 0,\ 0,\ 0,\ 0,\ 0,\text{-}3).$$
$$n_1 = (\ 1,\ 1,\ 0,\ 0,\text{-}1,\ 0,\ 0,\ 0,\text{-}3,\ 0,\ 0,\ 1).$$

3.1 The Existence of $\mathbf{JSF}_3$ for Pairs of Integers

Algorithm 1. ($\mathbf{JSF}_3$)

Input: Nonnegative integers n_0, n_1, not both zero.
Output: $\mathbf{JSF}_3$ for integers n_0, n_1
$n_0 = (u_{0,m-1}, \cdots, u_{0,1}, u_{0,0})$
$n_1 = (u_{1,m-1}, \cdots, u_{1,1}, u_{1,0}), u_{i,j} \in \{0, \pm 1, \pm 3\}, i = 0, 1, 0 \leq j < m.$

Set $k_0 \leftarrow n_0,\ k_1 \leftarrow k_1$
Set $j \leftarrow 0$
While $k_0 > 0$ or $k_1 > 0$ do
  For i from 0 to 1 do
    If k_i is even, then $u \leftarrow 0$
    Else
      $u \leftarrow$FOV(k_i)
      If k_{1-i} is even, then
        If k_{1-i}mod 8=4, then $u \leftarrow$FAV(k_i)
        If k_{1-i}mod 4=2 and k_i mod 32=$\pm 1, \pm 3$, then $u \leftarrow$ FNV(k_i)
        If k_{1-i}mod 4=2 and k_i mod 32=$\pm 5, \pm 11$, then $u \leftarrow$ FSV(k_i)
        If k_{1-i}mod 4=2 and k_i mod 32=$\pm 13, \pm 15$, then $u \leftarrow$ FSV(k_i)
        If k_{1-i}mod 32=$\pm 2, \pm 6$ and $k_i$mod 32=± 7, then $u \leftarrow$FSV(k_i)
        If k_{1-i}mod 32=$\pm 2, \pm 6$ and $k_i$mod 32=± 9, then $u \leftarrow$FNV(k_i)
        If k_{1-i}mod 32=$\pm 10, \pm 14$ and $k_i$mod 32=± 7, then $u \leftarrow$FNV(k_i)
        If k_{1-i}mod 32=$\pm 10, \pm 14$ and $k_i$mod 32=± 9, then $u \leftarrow$FSV(k_i)
      Else
        If k_i mod 32=$\pm 13, \pm 15$ and k_{1-i} mod 16=$\pm 5, \pm 7$,
        then $u \leftarrow$FAV(k_i)
        End If
        If k_i mod 16=$\pm 5, \pm 7$ and k_{1-i} mod 32=$\pm 13, \pm 15$,
        then $u \leftarrow$FAV(k_i)

End If
End If
End If
Set $u_{i,j} \leftarrow u$
Next i
Set $k_0 \leftarrow (k_0 - u_{0,j})/2, k_1 \leftarrow (k_1 - u_{1,j})/2$
Set $j \leftarrow j + 1$
EndWhile

In order to prove the desired properties of JSF_3, it is necessary to generalize *Alg*.1 by allowing as inputs JRF_3 for pairs of e_0, e_1.

Algorithm 2. ($\mathbf{JSF}_3$)

Input: $\mathbf{JRF}_3$ for integers e_0, e_1, not both zero.
$e_0 = (e_{0,m-1}, \cdots, e_{0,1}, e_{0,0})$
$e_1 = (e_{1,m-1}, \cdots, e_{1,1}, e_{1,0}), e_{i,j} \in \{0, \pm 1, \pm 3\}.i = 0, 1, 0 \leq j < m.$
Output: $\mathbf{JSF}_3$ for integers e_0, e_1

Set $j \leftarrow 0$
Set $d_0 \leftarrow 0, d_1 \leftarrow 0$
Set $u_{0,-2} \leftarrow 0, u_{0,-1} \leftarrow 0, u_{1,-2} \leftarrow 0, u_{1,-1} \leftarrow 0$
Set $a_0 \leftarrow e_{0,0}, b_0 \leftarrow e_{0,1}, x_0 \leftarrow e_{0,2}, y_0 \leftarrow e_{0,3}, z_0 \leftarrow e_{0,4}$
Set $a_1 \leftarrow e_{1,0}, b_1 \leftarrow e_{1,1}, x_1 \leftarrow e_{1,2}, y_1 \leftarrow e_{1,3}, z_1 \leftarrow e_{1,4}$
Set $k_0 \leftarrow a_0 + 2b_0 + 4x_0 + 8y_0 + 16z_0$
Set $k_1 \leftarrow a_1 + 2b_1 + 4x_1 + 8y_1 + 16z_1$

While $k_0 > 0$ or $k_1 > 0$ do
For i from 0 to 1 do
If k_i is even then $u \leftarrow 0$
Else **{SIMILAR TO Alg.1}**
End If
Set $u_{i,j} \leftarrow u$
Set $\beta_{i,j} \leftarrow (u_{i,j-2}, u_{i,j-1}, d_i, e_{i,j}, e_{i,j+1}, e_{i,j+2}, e_{i,j+3}, e_{i,j+4})$
Next i
Set $S_j \leftarrow (\beta_{0,j}, \beta_{1,j})$
Set $d_0 \leftarrow (d_0 + a_0 - u_{0,j})/2, d_1 \leftarrow (d_1 + a_1 - u_{1,j})/2$
Set $a_0 \leftarrow b_0, b_0 \leftarrow x_0, x_0 \leftarrow y_0, y_0 \leftarrow z_0, z_0 \leftarrow e_{0,j+5}$
Set $a_1 \leftarrow b_1, b_1 \leftarrow x_1, x_1 \leftarrow y_1, y_1 \leftarrow z_1, z_1 \leftarrow e_{1,j+5}$
Set $j \leftarrow j + 1$(if $j > m$, let $e_{i,j} = 0$)
Set $k_0 \leftarrow d_0 + a_0 + 2b_0 + 4x_0 + 8y_0 + 16z_0$
Set $k_1 \leftarrow d_1 + a_1 + 2b_1 + 4x_1 + 8y_1 + 16z_1$
EndWhile

The most straightforward way to prove the existence of JSF_3 for every pair of positive integers n_0, n_1 is to present an algorithm to produce it.

It is easy to check that, in the special case in which the $e'_{i,j}s$ are "ordinary" unsigned bits, $Alg.2$ is equivalent to $Alg.1$. So the correctness of the $Alg.2$ insures that of the $Alg.1$.

We call the vectors S_j the states of the algorithm, The output vector $(u_{0,j}, u_{1,j})$ is a function of the state S_j. Thus we may describe the action of $Alg.2$ as follows: that j^{th} iteration of the Do loop inputs the state S_{j-1}, outputs $(u_{0,j-1}, u_{1,j-1})$ and changes the state to S_j, namely,

$$S_{j-1} \xrightarrow{(u_{0,j-1}, u_{1,j-1})} S_j.$$

Let $\mathrm{t}_{i,j} = d_i + e_{i,j} + 2e_{i,j+1} + 4e_{i,j+2} + 8e_{i,j+3} + 16e_{i,j+4}$.

We next enumerate the possible values for the state and all the states are divided into below 12 cases based on the difference of S_j.

Table 1. State-Table

S_j	$\beta_{i,j}$	$\beta_{1-i,j}$
B_0	$\mathrm{t}_{i,j} \equiv 0 \bmod 16$	$\mathrm{t}_{1-i,j} \equiv 0 \bmod 16$
B_1	$\mathrm{t}_{i,j} \equiv 8 \bmod 16$	$\mathrm{t}_{1-i,j} \equiv 0 \bmod 8$
B_2	$\mathrm{t}_{i,j} \equiv 4 \bmod 8$	$\mathrm{t}_{1-i,j} \equiv 0 \bmod 8$
B_3	$\mathrm{t}_{i,j} \equiv 4 \bmod 8$	$\mathrm{t}_{1-i,j} \equiv 4 \bmod 8$
B_4	$\mathrm{t}_{i,j} \equiv 2 \bmod 4$	$\mathrm{t}_{1-i,j} \equiv 0 \bmod 4$
B_5	$\mathrm{t}_{i,j} \equiv 2 \bmod 4$	$\mathrm{t}_{1-i,j} \equiv 2 \bmod 4$
B_6	$\mathrm{t}_{i,j} \equiv 1 \bmod 2$	$\mathrm{t}_{1-i,j} \equiv 0 \bmod 2$
B_7	$\mathrm{t}_{i,j} \equiv \pm 13, \pm 15 \bmod 32$	$\mathrm{t}_{1-i,j} \equiv \pm 5 \bmod 16$
B_8	$\mathrm{t}_{i,j} \equiv \pm 13, \pm 15 \bmod 32$	$\mathrm{t}_{1-i,j} \equiv \pm 1, \pm 3 \bmod 16$
B_9	$\mathrm{t}_{i,j} \equiv \pm 1, \pm 3 \bmod 32$	$\mathrm{t}_{1-i,j} \equiv \pm 1, \pm 3 \bmod 32$
B_{10}	$\mathrm{t}_{i,j} \equiv \pm 1, \pm 3 \bmod 32$	$\mathrm{t}_{1-i,j} \equiv \pm 5, \pm 7 \bmod 16$
B_{11}	$\mathrm{t}_{i,j} \equiv \pm 5, \pm 7 \bmod 16$	$\mathrm{t}_{1-i,j} \equiv \pm 5, \pm 7 \bmod 16$

Table 2. State-Following-Table

S_j	S_{j+1}	S_j	S_{j+1}
B_0	B_0, B_1	B_6	$B_j, j \neq 4, j \neq 6$
B_1	B_2, B_3	B_7	B_5
B_2	B_4	B_8	B_1
B_3	B_5	B_9	B_0
B_4	B_6	B_{10}	B_2
B_5	$B_j, j = 7, \cdots, 11,$	B_{11}	B_3

It is easy to verify the following by checking all the cases. As a result, we have the following values for S_{j+1} for each S_j. All following states are shown in Table 2.

Theorem 1. *Alg.*1 always outputs the Width–3 Joint Sparse Form for its inputs. *Proof*. It is straightforward to verify that the expansion produced by the *Alg.*2 is in fact JGF for n_0, n_1. It remains to prove that this expansion satisfies terms of **Definition 2.** The process is similar to that [25]. And the proof appears in the Appendix A of the paper.

3.2 Efficiency of $\mathbf{JSF_3}$ for Pairs of Integers

Now, Our primary task is to prove the AJHD of JSF_3 is 37.1%.

It is easy to see that GF_3 is at most one bit longer than the ordinary binary expansion. As a result, JSF_3 is at most one bit longer than the binary expansion of the larger of the two integers.

Theorem 2. The average joint Hamming density among Joint 3–Sparse Form representations is 37.1%.

Proof. Let state space

$$\Gamma = \{G_i | i = 0, 1, \cdots, 10, 11\}, \text{ where } G_i = B_i,\ i = 0, 1, \cdots, 10, 11.$$

We can prove that a stochastic process $\{S_n | n \geq 0\}$ output by *Alg.*2 takes values in a countable set Γ and is a homogeneous Markov Chain in terms of Γ [see definition in page 252 [9]]. So, let $p_{i,j}$ denote the *transition probabilities* $p_{i,j}(n)$, where $p_{i,j}(n) = P\{S_{n+1} \in G_j | S_n \in G_i\}$. $\{p_{i,j}\}$ forms the following transition matrix P.

$$\mathrm{P} = \begin{bmatrix}
\frac{1}{4} & 0 & \frac{3}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{3}{16} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{64} & \frac{3}{64} & \frac{1}{8} & \frac{1}{16} & 0 & \frac{1}{4} & 0 & \frac{3}{32} & \frac{15}{128} & \frac{5}{32} & \frac{3}{32} & \frac{5}{32} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

From transition matrix P, for any two states $G_i, G_j \in \Gamma$, the state G_i is equivalent to G_j, so $\{S_n | n \geq 0\}$ is irreducible, and for any G_j, it is nonrecurrent. Therefore, the chain exists *stationary distrubution* $\{\pi_j, G_j \in \Gamma\}$, and $\lim_{m\to\infty}(\frac{1}{m}\sum_{n=1}^{m} p_{i,j}^{(n)}) = \pi_j$, where $p_{i,j}^{(n)} = P\{S_{(m+n)} \in G_j | S_m \in G_i\}$,$(G_i, G_j \in \Gamma, m \geq 0, n \geq 1)$.

From the equations below, which $\pi_j (j = 0, 1, \cdots, 11)$ satisfies [9],

$$(\pi_0, \pi_1, \cdots, \pi_{11}, 1) = (\pi_0, \pi_1, \cdots, \pi_{11})(\mathrm{P}, g^{\perp}).$$

where $g = (1,1,1,1,1,1,1,1,1,1,1,1)$, and the symbol $\perp$ denotes matrix transposition.
We get the solution

$$(\frac{4}{163}, \frac{12}{163}, \frac{20}{163}, \frac{16}{163}, \frac{20}{163}, \frac{61}{326}, \frac{20}{163}, \frac{19}{326}, \frac{129}{2608}, \frac{43}{2608}, \frac{19}{326}, \frac{43}{652}).$$

Let its absorbing probabilities $p_j(n) = p\{S_n \in G_j\}, j = 0, 1, \cdots, 11$, and initial distribution probabilities $p_j = p\{S_0 \in G_j\}, j = 0, 1, \cdots, 11$ of the chain, then the vector of $(u_{0,j}, u_{1,j}) = (0,0)$ is output by $G_j, j = 0, 1, \cdots, 5$. So AJHD is given by

$$\begin{aligned}\Sigma &= \sum_{j=6}^{11} \lim_{m\to\infty} \tfrac{1}{m} \sum_{n=1}^{m} p_j(n) \qquad = \sum_{j=6}^{11} \lim_{m\to\infty} (\tfrac{1}{m} \sum_{n=1}^{m} \sum_{G_i \in \Gamma} p_{i,j}^{(n)} p_i) \\ &= \sum_{j=6}^{11} \sum_{G_i} p_i (\lim_{m\to\infty} (\tfrac{1}{m} \sum_{n=1}^{m} p_{i,j}^{(n)})) = \sum_{j=6}^{11} \lim_{m\to\infty} (\tfrac{1}{m} \sum_{n=1}^{m} p_{i,j}^{(n)}) \\ &= \sum_{j=6}^{11} \pi_j.\end{aligned}$$

Therefore Σ =121/326. The AJHD of JSF_3 is 37.1% approximately.

4 Applications to ECC

The execution time of ECC schemes such as the ECDSA are typically dominated by point multiplications, In ECDSA, there are two types of point multiplications kP, where P is fixed (signature generation), and $uP + vQ$, where P is fixed and Q is not known a priori (signature verification). Using the above algorithm technique, the latter type can be sped by precomputation some data for points,such as $2P, 2Q, 3P, 3Q, P \pm Q, P \pm 3Q, 3P \pm Q, 3P \pm 3Q$, and storing some data for points such as $P, Q, 3P, 3Q, P \pm Q, P \pm 3Q, 3P \pm Q, 3P \pm 3Q$. Adapting the fast Straus' Method by using JSF_3 yields a technique which requires $\sim l$ doublings and $\sim (0.37)l$ general additions (on average). In other words, that sometimes works almost 8.6% faster than that by using the Joint Sparse Form.

The front type can also be sped. The following is a simplest approach. Suppose that the order r of the private key space is less than l. Let $Q = 2^{\lfloor l/2)+1 \rfloor} P$, then $k = a + b2^{\lfloor l/2)+1 \rfloor} P$, thus compute $k = aP + bQ$, one applies **Alg.1** to generate JSF_3 for integers a, b . This technique of computing it using JSF_3 requires $\sim l$ doublings and $\sim (0.19l)$ additions, which wins over that using JSF.

If the Elliptic Curves are particular curves, as Koblitz Curves, there may be the form with width–3, analogous to JSF_3. So, it would be of interest to construct the forms which apply to Koblitz Curves.

References

1. R. Avanzi. On Multi-exponentiation in Cryptography. 2003, manuscript, Available at http://citeseer.nj.nec.com/545130.html.
2. D. J. Bernstein. Pippenger'2 exponentiation algorithm. Available at: http://cr.rp.to/papers.html,2002.
3. M. Brown, D. Hankerson, J. Lopez and A. Menezes. Software Implementation of NIST Elliptic Curves Over Prime Fields. CACR Technical Reports. CORR 2000-56, University of Waterloo, 2000.
4. R. Crandall. Method and Apparatus for Public Key Exchange in a Cryptographic System. U.S. Patent # 5, 159, 632, Oct 27.1992.
5. H. Cohen. *A Course in Computational Algebraic Number Theory*, Volume 138 of Graduate Texts in Mathematics, Springer, 1996.
6. T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete logarithms. *IEEE Trans. on Information Theory* IT-31 (1985), pp. 469-472.
7. R. Gallant, R. Lambert and S. Vanstone. Faster Point Multiplication On Elliptic Curves with Efficient Endomorphisms. *Advances in Cryptology – Crypto 2001, LNCS*, Volume 2139, Springer-Verlag, pages 190-200, 2001.
8. D.M. Gordon. A Survey of Fast Exponentiation Methods. *Journal of Algorithms*, 27(1):129-146,1998.
9. KaiLai Chung. *Elementary Probability Theory with Stochastic Processes*. Springer-Verlag Berlin Heidelberg New York Toppan Company (S)Pte Ltd. Singapore 1978.
10. IEEEP 1363-2000,(2000). *IEEE standard Specifications for Public-Key Cryptography*. IEEE Computer Society, August 29,2000.
11. D.E. Knuth. *The Art of Computer Programming*. Vol 2: Semi numerical Algorithms 2nd ed., Addison-Wesley,1981.
12. Mathieu Ciet, Tanja Lange. Franceso Sica, and Jean-Jacques Quisquater. Improved Algorithms for Efficient Arithmetic on Elliptic Curves using Fast Endomorphisms. *Advances in Cryptology – Eurocrypt 2003, LNCS*, Volume 2656, Springer-Verlag, pages 388-400, 2003.
13. Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(1987), 203-209.
14. Neal Koblitz. CM-Curves with Good Cryptographic Properties. *Advances in Cryptology – Crypto 91, LNCS*, Volume 576, Springer-Verlag, pages 279-287, 1992.
15. V.Miller. Uses of elliptic curves in cryptography. *Advances in Cryptology – Crypto 85, LNCS*, Volume 218, Springer-Verlag, pages 417-426, 1986.
16. F.Morain and J.Olivos. Speeding Up the Computations on an Elliptic Curve Using Addition-Subtraction Chains. *Inform.Theor. Appl.*, Volume 24, pages 531-543, 1990.
17. Willi Meier, Othmar Staffelbach. Efficient Multiplication on Certain Nonsupersingular Elliptic Curves. *Advances in Cryptology – Crypto 92, LNCS*, Volume 740, Springer-Verlag, pages 333-344, 1993.
18. V.Muller. Fast Multiplication on Elliptic Curves over Small Fields of Characteristic Two. *Journal of Cryptology*, 11(4):219-234, 1998.
19. V.Muller. Efficient Point Multiplication for Elliptic Curves over Special Optimal Extension Fields. In Walter de Gruyter, Editor, *Public-Key Cryptography and Computational Number Theory*, pages 197-207, Warschau, Poland, September 11-15, 2000(2001).
20. National Institute of Standards and Technology. *FIPS – 186-2: Digital Signature Standard (DSS)*, January 2000. Available at http://csrc.nist.gov/publications/fips.

21. Y-H. Park, S. Jeong, C. Kim, and J. Lim. An Alternate Decomposition of an Integer for Faster Point Multiplication on Certain Elliptic Curves. *Advances in Crytpology – PKC 2002, LNCS*, Volume 2274, Springer-Verlag, pages 323-334, 2002.
22. J.H. Silverman. *The Arithmetic of Elliptic Curves*, GTM 106, Springer-Verlag, 1986.
23. J. Solinas. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. *Advances in Cryptology – Crypto 1997, LNCS*, Volume 1294, Springer-Verlag, pages 357-371, 1997.
24. J. Solinas. Efficient Arithmetic on Koblitz Curves. *Designs, Codes and Cryptography*, 19:195-249, 2000.
25. J. Solinas. Low-Weight Binary Representations for Pairs of Integers. CACR Technical Reports, CORR 2001-41 University of Waterloo, 2001, Available at: www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-41.ps, 2001.
26. Yasuyuki Sakai and Kouichi Sakurai. Algorithms for Efficient Simultaneous Elliptic Scalar Multiplication with Reduced Joint Hamming Weight Representation of Scalars, *5th International Conference, ISC 2002, LNCS*, Volume 2443, Springer-Verlag, pages 484-499, 2002.
27. N.P. Smart. Elliptic Curve Cryptosystems over Small Fields of Odd Characteristic. *Journal of Cryptology*, 12(2):141-151, 1999.

Appendix A: The Proof of Theorem 1

Theorem 1. *Alg.*1 always produces the width–3 joint sparse form expression of its inputs.

Proof: It is straightforward to verify that the expansion produced by the *Alg.*2 is in fact JGF for n_0, n_1. It remains to prove that this expansion satisfies properties (JSF_3–1.), (JSF_3–2.), (JSF_3–3.), (JSF_3–4). From the **Table 1** and **Table 2**, The process that proves the conclusions follows as:

(JSF_3–1.): This condition is equivalent to the assertion that, for every j, at least one of S_j, S_{j+1}, S_{j+2} is in one of states B_i, $i = 0, \cdots, 5$, and at least two of $S_j, S_{j+1}, S_{j+2}, S_{j+3}, S_{j+4}$ are in the states B_i, $i = 0, \cdots, 5$. Firstly, we prove that for every j, at least one of S_j, S_{j+1}, S_{j+2} is in one of states B_i, $i = 0, \cdots, 5$. Suppose that S_j isn't in any $B_i, i = 0, \cdots, 5$, then S_j is in states $B_i, i = 6, \cdots, 11$. If S_j is in states B_6, then S_{j+1} or S_{j+2} is in one of states B_i, $i = 0, \cdots, 5$; if S_j is in one of states $B_i, i = 7, \cdots, 11$, then S_{j+1} is in one of states $B_i, i = 1, \cdots, 5$. So S_{j+2} or S_{j+3} is in one of states B_i, $i = 0, \cdots, 5$. Secondly, the process that proves at least two of $S_j, S_{j+1}, S_{j+2}, S_{j+3}, S_{j+4}$ are in the states B_i, $i = 0, \cdots, 5$ is similar to the above.

(JSF_3–3.): Might as well, suppose that $u_{0,j} \neq 0, u_{0,j+1} \neq 0$, then it follows from the Table 2 that S_j is in the states B_6 and S_{j+1} is in one of states $B_i, i = 7, \cdots, 11$. It is straightforward to compute and to verify that $u_{1,j+1} \neq 0$, $u_{1,j} = 0$.

(JSF_3–4.): Might as well, suppose that $u_{0,j} \neq 0, u_{0,j+2} \neq 0$, then S_j is in one of states B_6, B_7, and S_{j+1} is in the state B_5, and S_{j+2} is in the state $B_i, i = 7, \cdots, 11$. It is straightforward to compute and to verify that $u_{1,j+2} \neq 0$.

(JSF_3–2.): Might as well, suppose that $u_{0,j} \neq 0, u_{0,j+1} \neq 0$, then S_j is in the states B_6 and S_{j+1} is in one of states $B_i, i = 7, \cdots, 11$. Suppose $u_{0,j} \cdot u_{0,j+1} = -1$,

then $u_{0,j} = 1, u_{0,j+1} = -1$, or $u_{0,j} = -1, u_{0,j+1} = 1$, $t_{0,j}$ mod 8=± 1. So $u_{0,j}$ only fetches FSV ($t_{0,j}$), and $t_{0,j} = \pm 1, \pm 7, \pm 9, \pm 15$ mod 32. Thus, according to the **Alg.1** the conditions that may be satisfies $u_{0,j} \cdot u_{0,j+1} = -1$ shown as following,

(1.) $t_{0,j} = \pm 7$ mod 32 and $t_{1,j} = \pm 2, \pm 6$ mod 32.
(2.) $t_{0,j} = \pm 9$ mod 32 and $t_{1,j} = \pm 10, \pm 14$ mod 32.
(3.) $t_{0,j} = \pm 15$ mod 32 and $t_{1,j} = \pm 2, \pm 6$ mod 16.

If $t_{0,j} = \pm 7$ mod 32 and $t_{1,j} = \pm 2, \pm 6$ mod 32, then $u_{0,j} = \pm 1, u_{0,j+1} = \pm 3$, so it is not correct.

Similarly, if $t_{0,j} = \pm 9$ mod 32 and $t_{1,j} = \pm 10, \pm 14$ mod 32, then $u_{0,j} = \pm 1, u_{0,j+1} = \pm 3$, so it is not correct.

Similarly, if $t_{0,j} = \pm 15$ mod 32 and $t_{1,j} = \pm 2, \pm 6$ mod 16, then $u_{0,j} = \pm 1, u_{0,j+1} = \pm 3$, so it is also not correct.

Therefore, there is not the condition which satisfies the $u_{0,j} \cdot u_{0,j+1} = -1$. Namely, $u_{0,j} \cdot u_{0,j+1} \neq -1$.