ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.9, September 2006, pp.2004−2012
DOI: 10.1360/jos172004

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

整数对的低重量表示 JSF_3*

张亚娟+, 祝跃飞, 况百杰

(解放军信息工程大学 信息工程学院 网络工程系,河南 郑州 450002)

Low-Weight JSF_3 Representations for Pairs of Integers

ZHANG Ya-Juan+, ZHU Yue-Fei, KUANG Bai-Jie

(Department of Network Engineering, Information Engineering University, Zhengzhou 450002, China)

+ Corresponding author: Phn: +86-371-63530540, E-mail: springzyj@yahoo.com.cn

Zhang YJ, Zhu YF, Kuang BJ. Low-Weight JSF_3 representations for pairs of integers. *Journal of Software*, 2006,17(9):2004−2012. http://www.jos.org.cn/1000-9825/17/2004.htm

Abstract: J.A.Solinas suggested an optimal signed binary representation for pairs of integers, which is called a Joint Sparse Form (JSF). 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 a new representations, for pairs of integers, which is called Width-3 Joint Sparse Form (JSF_3). The representation is at most one bit longer than the binary expansion of the larger of the two integers, and the average joint Hamming density is 19/52. So, computing the form of $uP+vQ$ by using JSF_3 is almost 9% faster than that by using JSF.
Key words: elliptic curve cryptosystem; ECDSA; JSF; width-3 joint sparse form (JSF_3); AJHD

摘 要: J.A.Solinas 给出了整数对的最优带符号二进制表示,称做联合稀疏表示(JSF).JSF 表示长度至多是最大整数的二进制长度加一,其平均汉明密度为 1/2.利用窗口方法扩展了联合稀疏表示,给出了整数对的一种新表示方法:3-宽度联合稀疏表示(JSF_3).该表示长度至多是最大整数的二进制长度加一,平均汉明密度为 19/52.因此,利用 JSF_3 计算 $uP+vQ$ 比用 JSF 大约提高 9%的效率.
关键词: 椭圆曲线密码;ECDSA;JSF;3-宽度联合稀疏表示(JSF_3);AJHD
中图法分类号: TP309 文献标识码: A

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 $GF(q)$ is to compute g^a for a given element $g \in GF(q)$ and a positive integer a. This is typically accomplished by the binary method, based on the binary expansion of a.

* Supported by the National Natural Science Foundation of China under Grant No.90204015 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.G1999035804 (国家重点基础研究发展计划(973)); the Elitist Youth Foundation of Henan Province under Grant No.021201400 in China (河南省杰出青年基金)

Received 2003-11-03; Accepted 2005-07-28

The method requires approximately $l/2$ general multiplications and approximately l squarings (on average) ($l=\lceil \log_2 q \rceil$).

More generally, it is commonly 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[1] (also called Squaring-Multiple Method), presented by Shamir in 1985. The method requires proximately l general multiplications and proximately l squarings (on average). After then, numerous methods for speeding up scalar multiplication have been discussed in the literature; for a survey, see Ref.[2].

While on general Elliptic Curve $E(GF(q))$, $P=(x,y)\in E(GF(q))$, then $-P=(x,-y)$. Thus point subtraction is as efficient as addition. This motivates the use of a signed binary expansion (allowing coefficients 0 and ±1). A particularly useful signed digit representation is the non-adjacent form (NAF)[3]. By using a window method, one processes some other signed digit representation, called the width-w nonadjacent form (NAF_w)[2−4]. (when w=2, NAF_w is equivalent to NAF). There is a simple and efficient algorithm for presenting NAF_w of any integer. When Computing kP, the method requires approximately $l/(w+1)$ general point addition and l double.

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. In the following, we will call this form as multi scalar multiplications. So the efficiency of implementation depends mostly on the efficiency of evaluation of the multi scalar multiplications. Thus, fast multi scalar multiplication is essential for Elliptic Curve Cryptosystems. There are lots of research papers on the problem of speeding up $uP+vQ$ in recent years[2−7].

For computing the form $uP+vQ$, J.A.Solinas suggested an optimal signed binary representation for pairs of integers, called Joint Sparse Form (JSF). 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 presents the concept of form representation of integers, brings forward Width-3 Joint Sparse Form (JSF_3), extends the JSF method by using some other signed digit representation of integers, and also proves that the average joint Hamming density(AJHD) is 19/52. So, this improvement can speed up the computation of the form $uP+vQ$ by up to 9%, while compared to computation by using JSF.

The paper is organized as follows: Section 2 gives 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 unique existence, and presents an algorithm for producing it, and finally shows AJHD of that is 19/52 via stochastic process; Section 4 gives the application of the technique and discusses the avenues for further work.

2 Preparation Knowledge

A given nonegative integer n has a common binary expansion $n=(a_l,\ldots,a_1,a_0)=\sum_{i=0}^{l} a_i 2^i$,a_i=0,1. It has another binary expansion $n=(b_t,\ldots,b_1,b_0)=\sum_{i=0}^{t} b_i 2^i$, $b_i\in\{0,\pm1,\pm3,\ldots,\pm(2^{w-1}-1)\}$, $w>0$. We call it the width-w generalized (binary) expansion form of a (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 a unique NAF_w[3]. 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 $l/(w+1)$[3].

Definition 1. A width-3 generalized expansion of n is half sparse form (HSF_3) if it satisfies the following conditions:

1. Of any four consecutive terms, at least two are zero;
2. The product of any adjacent terms equals to 0, 3, 9.

Let n be a positive integer, then the notation "n mods 8" indicates that the modular reduction 8 is to return the smallest residue in absolute value. Correspondingly for Width-3 generalized expansions of u, $u=(a_l,\ldots,a_1,a_0)$, obviously, $a_0=0$ if n is an even number; and if n is an odd number, then $a_0\in\{n \text{ mods } 8, (n+4) \text{ mods } 8, -(n \text{ mods } 8), -((n+4) \text{ mods } 8)\}$. So, we may call a_0

1. Fetching-Original-Value of n ($FOV(n)$), if $a_0=n$ mods 8;
2. Fetching-Anti-Value of n ($FAV(n)$), if $a_0=(n+4)$ mods 8;
3. Fetching-Sign-Value of n ($FSV(n)$), if $a_0=-(n \text{ mods } 8)$;
4. Fetching-Number-Value of n ($FNV(n)$), if $a_0=-((n+4) \text{ mods } 8)$.

Lemma 1. For HSF_3 for n, $n=(a_l,\ldots,a_1,a_0)$, we can obtain that:

1. a_0 only equals to $FOV(n)$, $FAV(n)$, if $n=\pm1, \pm3 \bmod 16$;
2. a_0 may equal to $FOV(n)$, $FAV(n)$, $FSV(n)$, if $n=\pm5 \bmod 16$;
3. a_0 may fetch $FOV(n)$, $FAV(n)$, $FSV(n)$, $FNV(n)$, if $n=\pm7 \bmod 16$.

For any four consecutive terms $(a_{j+3},a_{j+2},a_{j+1},a_j)$, we use the sign of "0" as the value that is certain to be zero, and use the sign of "*" as the value that is certain to be nonzero, and use the sign of "?" as the value that is not certain. The pattern denoted by the signs of "0", "*", "?" is called pattern of a_j. It is easy to follow that if the expansion for n is HSF, when a_j fetches FOV, FAV, FSV, FNV, then its pattern corresponds to the form of (?,0,0,*),(0,*,0,*),(0,0,*,*),(0,0,*,*).

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). Furthermore, we call it the reduced width-3 generalized expansion form of n_0, n_1 (JRF_3) if both are reduced. Analogically, we call it the joint NAF_3 (JNF_3). The number of the nonzero columns of JGF_3 is called the joint Hamming weight (JHW) and the ratio of JHW to its length is its average joint Hamming density (AJHD), where n_0, n_1 run over l-bit integers of N. It isn't difficult to see that JHW of JNF_3 is quite smaller among all JGF_3s, but it is not the smallest. Thereinafter, we give the expansion that is the smallest among JGF_3s, whose AJHD is 19/52, while that of JNF_3 is 7/16.

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

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

is called Width-3 Joint Sparse Form ($JSF_3(n_0,n_1)$), shortly noted by JSF_3, if the expansion satisfies the following conditions:

1. JSF_3-1: Of any four consecutive columns, at least two are zeros;
2. JSF_3-2: For every row, the product of the adjacent terms is 0, 3, 9;
3. JSF_3-3: There are five pieces of status for interconnect of two rows:
   a) If there exists $i\in\{0,1\}$ which satisfies $u_{i,j}u_{i,j+1}=3,9$, i.e. $u_{i,j}\neq0$, $u_{i,j+1}\neq0$, then $u_{1-i,j}=0$, $u_{1-i,j+1}\neq0$.
   b) If $u_{0,j}$, $u_{1,j}$ are not both zero, then $u_{0,j+2}=0$, $u_{1,j+2}=0$, or $u_{0,j+2}\neq0$, $u_{1,j+2}\neq0$. Furthermore, if $u_{0,j-2}$, $u_{0,j-1}$, $u_{1,j-2}$, $u_{1,j-1}$ are not all zero, then $u_{0,j+2}=0$, $u_{1,j+2}=0$.

c) If $u_{0,j}\neq 0$, $u_{0,j+2}\neq 0$, $u_{1,j}\neq 0$, $u_{1,j+2}\neq 0$, then there exists $i\in\{0,1\}$ which satisfies $u_{i,j}u_{i,j+2}>0$, $u_{i,j+2}=\pm 3$ and $u_{1-i,j}u_{1-i,j+2}>0$, $u_{1-i,j+2}=\pm 1$, or $u_{1-i,j}u_{1-i,j+2}<0$, $u_{i,j+2}=\pm 3$.

d) If $u_{0,j-2}$, $u_{0,j-1}u_{1,j-2}u_{1,j-1}$ are all zero and $u_{0,j}\neq 0$, $u_{1,j}\neq 0$, then $u_{0,j+3}u_{1,j+4}=u_{0,j+4}u_{1,j+3}=0$.

e) If there exists $i\in\{0,1\}$ which satisfies $u_{i,j}\neq 0$, $u_{i,j+1}=\pm 3$, then $u_{0,j+4}=0$, $u_{1,j+4}=0$, or $u_{0,j+4}\neq 0$, $u_{1,j+4}\neq 0$.

Obviously, each row expansion of JSF_3 is HSF_3 of an integer.

3.1 Uniqueness of JSF_3 for pairs of integers

Theorem 1. A pair of positive integers has at most one JSF_3.

Proof: Suppose, on the contrary, that there are two distinct JSF_3:

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

Since these representations are different, then $u_{i,j}\neq w_{i,j}$ for some i,j. Let g be the minimal value of j for which the two forms disagree. For i=0,1, set $k_i=(u_{i,m-1},...,u_{i,g+1},u_{i,g})=(w_{i,n-1},\ldots,w_{i,g+1},w_{i,g})$. Since the expansions disagree at $j=g$, we may assume that $u_{0,g}\neq w_{0,g}$ by exchanging the roles of n_0, n_1 if necessary. It follows that k_0 is odd, for otherwise we would have $u_{0,g}=w_{0,g}=0$. Therefore $u_{0,g}$, $w_{0,g}$ must have values ±1, ±3. Then it follows from Definition 2 that

$$(u_{0,g+3},u_{0,g+2},u_{0,g+1},u_{0,g})=(?,0,0,*),(0,*,0,*),(0,0,*,*),$$
$$(w_{0,g+3},w_{0,g+2},w_{0,g+1},w_{0,g})=(?,0,0,*),(0,*,0,*),(0,0,*,*),$$

and when $u_{0,g}$, $w_{0,g}$, fetch *FOV*, *FAV*, *FSV*, *FNV*, $u_{1,g}$, $w_{1,g}$ also correspond to only fetching *FOV*, *FAV*, *FSV*, *FNV* if k_0, k_1 are both odd.

We can prove that the assumption is not correct by analyzing the four cases of $u_{0,g}$, $w_{0,g}$. Because the space is limited, the details are omitted.

3.2 The existence of JSF_3 for pairs of integers

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 for producing it.

Algorithm 1. **JSF_3**.

Input: Nonnegative integers n_0,n_1, not both zero.

Output: JSF_3 for integers n_0, n_1.

$n_0=(u_{0,m-1},\ldots,u_{0,1},u_{0,0})$

$n_1=(u_{1,m-1},\ldots,u_{1,1},u_{1,0}),u_{i,j}\in\{0,\pm 1,\pm 3\}$, i=0,1, $0\leq j\leq m$.

1. Set $k_0\leftarrow n_0$, $k_1\leftarrow n_1$;
2. Set $j\leftarrow 0$;
3. Set $u_{0,-2}\leftarrow 0$, $u_{0,-1}\leftarrow 0$, $u_{1,-2}\leftarrow 0$, $u_{1,-1}\leftarrow 0$;
4. 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

   $u\leftarrow k_i$ mods 8

   If $k_i\equiv\pm 1,\pm 3$ mod 16 then

   If $k_{1-i}\equiv 4$ mod 8 then u$\leftarrow(u+4)$ mods 8

   If $k_{1-i}\equiv\pm 5,\pm 7$ mod 16 and $k_i\equiv\pm 13,\pm 15$ mod 32 and

   $u_{0,j-2}=u_{0,j-1}=u_{1,j-2}=u_{1,j-1}=0$ then $u\leftarrow(u+4)$ mods 8

   EndIf

Elseif $k_i \equiv \pm 5 \bmod 16$ then
If $k_{1-i} \equiv 4 \bmod 8$ then u←$(u+4)$ mods 8
If $k_{1-i} \equiv \pm 2 \bmod 8$ then u←$-u$
If $k_{1-i} \equiv \pm 13, \pm 15 \bmod 32$ and $u_{0,j-2}=u_{0,j-1}=u_{1,j-2}=u_{1,j-1}=0$ then
u←$(u+4)$ mods 8
Endif
Elseif $k_i \equiv \pm 7 \bmod 16$ then
If $k_{1-i} \equiv 4 \bmod 8$ then u←$(u+4)$ mods 8
If $k_{1-i} \equiv \pm 2 \bmod 8$ then
a_0←$((k_0>>3)+(k_0>>4)) \bmod 2$
a_1←$((k_1>>3)+(k_1>>4)) \bmod 2$
If $a_0=a_1$ then u←$-u$
else u←$-((u+4)$ mods 8)
Endif
Endif
If $k_{1-i} \equiv \pm 13, \pm 15 \bmod 32$ and $u_{0,j-2}=u_{0,j-1}=u_{1,j-2}=u_{1,j-1}=0$ then
u←$(u+4)$ mods 8
Endif
Endif
Endif
Set $u_{i,j}$←u;
Next i;
Set k_0←$(k_0-u_{0,j})/2$, k_1←$(k_1-u_{1,j})/2$;
Set j←$j+1$;
EndWhile

In order to prove the desired properties of JSF_3, it is necessary to generalize Algorithm 1 by allowing inputs JRF_3 for a pair of e_0, e_1.

Algorithm 2. **JSF_3**.

Input: JRF_3 for integers e_0,e_1, not both zero.

$e_0=(e_{0,m-1},\ldots,e_{0,1},e_{0,0})$,

$e_1=(e_{1,m-1},\ldots,_1,e_{1,0})$. $e_{i,j} \in \{0,\pm 1,\pm 3\}$, i=0,1, $0 \le j \le$m.

Output: JSF_3 for e_0, e_1

1. Set j←0;
2. Set d_0←0, d_1←0;
3. Set $u_{0,-2}$←0, $u_{0,-1}$←0, $u_{1,-2}$←0, $u_{1,-1}$←0;
4. Set a_0←$e_{0,0}$, b_0←$e_{0,1}$, x_0←$e_{0,2}$, y_0←$e_{0,3}$, z_0←$e_{0,4}$;
5. Set a_1←$e_{1,0}$, b_1←$e_{1,1}$, x_1←$e_{1,2}$, y_1←$e_{1,3}$, z_1←$e_{1,4}$;
6. Set k_0←$a_0+2b_0+4x_0+8y_0+16z_0$;
7. Set k_1←$a_1+2b_1+4x_1+8y_1+16z_1$;
8. While $k_0>0$ or $k_1>0$ do
For i from 0 to 1 do
If k_i is even then u←0

Else SIMILAR TO Algorithm 1

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

It is easy to check that, in the special case in which the $e_{i,j}$'s are "ordinary" unsigned bits, Algorithm 2 is equivalent to Algorithm 1. So the correctness of the Algorithm 2 insures that of the Algorithm 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 Algorithm 2 as follows: the 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 states to S_j, namely, $S_{j-1}\xrightarrow{(u_{0,j-1},u_{1,j-1})}S_j$.

Let $\tau_{i,j}=d_i+e_{i,j}+2e_{i,j+1}+4e_{i,j+2}+8e_{i,j+3}+16e_{i,j+4}$. We next enumerate the possible values for the state and all the states are divided into the following 24 cases of the difference of S_j (see Tables 1 and 2).

Table 1 State-Table

S_j	$\beta_{0,j}$	$\beta_{1,j}$	Form-of-Fetch-Value
$B_{0,0}$	$t_{0,j}\equiv 0 \bmod 4$	$t_{1,j}\equiv 0 \bmod 4$	(0,0)
$B_{0,1}$	$t_{0,j}\equiv 4 \bmod 8$	$t_{1,j}\equiv 1 \bmod 2$	(0,*FAV*)
$B_{0,2}$	$t_{0,j}\equiv 0 \bmod 8$	$t_{1,j}\equiv 1 \bmod 2$	(0,*FOV*)
$B_{0,3}$	$t_{0,j}\equiv\pm 2 \bmod 8$	$t_{1,j}\equiv\pm 1, \pm 3 \bmod 16$	(0,*FOV*)
$B_{0,4}$	$t_{0,j}\equiv\pm 2 \bmod 8$	$t_{1,j}\equiv\pm 5 \bmod 16$	(0,*FSV*)
$B_{0,5}$	$t_{0,j}\equiv\pm 2,\pm 6 \bmod 32$	$t_{1,j}\equiv\pm 7 \bmod 32$	(0,*FSV*)
$B_{0,6}$	$t_{0,j}\equiv\pm 2,\pm 6 \bmod 32$	$t_{1,j}\equiv\pm 9 \bmod 32$	(0,*FNV*)
$B_{0,7}$	$t_{0,j}\equiv\pm 10,\pm 14 \bmod 32$	$t_{1,j}\equiv\pm 7 \bmod 32$	(0,*FNV*)
$B_{0,8}$	$t_{0,j}\equiv\pm 10,\pm 14 \bmod 32$	$t_{1,j}\equiv\pm 9 \bmod 32$	(0,*FSV*)
$B_{1,0}$	$t_{0,j}\equiv 0 \bmod 2$	$t_{1,j}\equiv 0 \bmod 2$	(0,0)(*1)
$B_{1,1}$	$t_{0,j}\equiv 1 \bmod 2$	$t_{1,j}\equiv 4 \bmod 8$	(*FAV*,0)
$B_{1,2}$	$t_{0,j}\equiv 1 \bmod 2$	$t_{1,j}\equiv 0 \bmod 8$	(*FOV*,0)
$B_{1,3}$	$t_{0,j}\equiv\pm 1,\pm 3 \bmod 16$	$t_{1,j}\equiv\pm 2 \bmod 8$	(*FOV*,0)
$B_{1,4}$	$t_{0,j}\equiv\pm 5 \bmod 16$	$t_{1,j}\equiv\pm 2 \bmod 8$	(*FSV*,0)
$B_{1,5}$	$t_{0,j}\equiv\pm 7 \bmod 32$	$t_{1,j}\equiv\pm 2,\pm 6 \bmod 32$	(*FSV*,0)
$B_{1,6}$	$t_{0,j}\equiv\pm 9 \bmod 32$	$t_{1,j}\equiv\pm 2,\pm 6 \bmod 32$	(*FNV*,0)
$B_{1,7}$	$t_{0,j}\equiv\pm 7 \bmod 32$	$t_{1,j}\equiv\pm 10,\pm 14 \bmod 32$	(*FNV*,0)
$B_{1,8}$	$t_{0,j}\equiv\pm 9 \bmod 32$	$t_{1,j}\equiv\pm 10,\pm 14 \bmod 32$	(*FSV*,0)
$B_{2,0}$	$t_{0,j}\equiv 1 \bmod 2$	$t_{1,j}\equiv 1 \bmod 2$	(*FOV*,*FOV*)(*2)
$B_{2,1}$	$t_{0,j}\equiv\pm 13,\pm 15 \bmod 32$	$t_{1,j}\equiv\pm 5,\pm 7 \bmod 16$	(*FAV*,*FAV*)(*3)
$B_{2,2}$	$t_{0,j}\equiv\pm 5,\pm 7 \bmod 16$	$t_{1,j}\equiv\pm 13,\pm 15 \bmod 32$	(*FAV*,*FAV*)(*3)
$B_{2,3}$	$t_{0,j}\equiv\pm 1,\pm 3 \bmod 32$	$t_{1,j}\equiv\pm 1 \bmod 2$	(*FOV*,*FOV*)(*3)
$B_{2,4}$	$t_{0,j}\equiv\pm 13,\pm 15 \bmod 32$	$t_{1,j}\equiv\pm 1,\pm 3 \bmod 16$	(*FOV*,*FOV*)(*3)
$B_{2,5}$	$t_{0,j}\equiv\pm 5,\pm 7 \bmod 16$	$t_{1,j}\not\equiv\pm 13,\pm 15 \bmod 32$	(*FOV*,*FOV*)(*3)

Table 2 State-Following-Table

S_j	$u_{0,j}$	$u_{1,j}$	S_{j+1}	S_j	$u_{0,j}$	$u_{1,j}$	S_{j+1}
$B_{0,0}$	0	0	$B_{0,0}$,$B_{1,0}$	$B_{1,0}$	0	0	(*4)
$B_{0,1}$	0	±1,±3	$B_{1,0}$	$B_{1,1}$	±1,±3	0	$B_{1,0}$
$B_{0,2}$	0	±1,±3	$B_{0,0}$	$B_{1,2}$	±1,±3	0	$B_{0,0}$
$B_{0,3}$	0	±1,±3	$B_{1,2}$	$B_{1,3}$	±1,±3	0	$B_{0,2}$
$B_{0,4}$	0	±3	$B_{2,0}$	$B_{1,4}$	±3	0	$B_{2,0}$
$B_{0,5}$	0	±1	$B_{2,0}$	$B_{1,5}$	±1	0	$B_{2,0}$
$B_{0,6}$	0	±3	$B_{2,0}$	$B_{1,6}$	±3	0	$B_{2,0}$
$B_{0,7}$	0	±3	$B_{2,0}$	$B_{1,7}$	±3	0	$B_{2,0}$
$B_{0,8}$	0	±1	$B_{2,0}$	$B_{1,8}$	±1	0	$B_{2,0}$
$B_{2,0}$	±1,±3	±1,±3	$B_{0,0}$	$B_{2,3}$	±1,±3	±1,±3	$B_{0,0}$
$B_{2,1}$	±1,±3	±1,±3	$B_{1,0}$	$B_{2,4}$	±1,±3	±1,±3	$B_{0,0}$
$B_{2,2}$	±1,±3	±1,±3	$B_{1,0}$	$B_{2,5}$	±1,±3	±1,±3	$B_{0,0}$

Note: (*1) means at least one isn't divisible by 4; (*2) means $u_{0,j-2}$, $u_{0,j-1}$, $u_{1,j-2}$, $u_{1,j-1}$ are not all zero; (*3) means $u_{0,j-2}$, $u_{0,j-1}$, $u_{1,j-2}$, $u_{1,j-1}$ are all zero, and $t_{1,j}\neq\pm13,\pm15 \bmod 32$ means $t_{1,j}=\pm1,\pm3 \bmod 32$, or $t_{1,j}=\pm5,\pm7 \bmod 16$; (*4) indicates that any state is a possible successor to state $B_{1,0}$, excluding $B_{0,0}$, $B_{1,0}$.

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 the following states are shown in Table 2. Finally, it will be useful to further combine the above 24 states into six as follows (see Table 3).

Table 3 Simple-State-Following-Table

S_j		$u_{0,j}=u_{1,j}=0$	S_{j+1}
C_0	$B_{0,0}$	YES	C_0,C_1
C_1	$B_{1,0}$	YES	C_2,C_3,C_4,C_5
C_2	$B_{0,1}$,$B_{1,1}$,$B_{2,1}$,$B_{2,2}$	NO	C_1
C_3	$B_{0,2}$,$B_{1,2}$,$B_{2,0}$	NO	C_0
C_4	$B_{0,3}$,$B_{0,4}$,$B_{0,5}$,$B_{0,6}$,$B_{0,7}$,$B_{0,8}$	NO	C_3
C_4	$B_{1,3}$,$B_{1,4}$,$B_{1,5}$,$B_{1,6}$,$B_{1,7}$,$B_{1,8}$	NO	C_3
C_5	$B_{2,3}$,$B_{2,4}$,$B_{2,5}$	NO	C_0

Theorem 2. Algorithm 1 always outputs the Width-3 Joint Sparse Form for its inputs.

Proof: It is straightforward to verify that the expansion produced by the Algorithm 2 is in fact JGF for n_0, n_1. It remains to prove that this expansion satisfies the terms of Definition 2. The process is similar to that.

3.3 Efficiency of JSF_3 for pairs of integers

Now, Our primary task is to prove that AJHD of JSF_3 is 19/52. 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 3. The average joint Hamming density among Joint 3-Sparse Form representations is 19/52.

Proof: Let state space $\Gamma=\{G_i|i=0,1,\ldots,10,11\}$, where,

$G_0=\{S_n\in C_0|S_{n-1}\in C_5\}$, $G_1=\{S_n\in C_0|S_{n-2}\in C_4\}$ $G_2=\{S_n\in C_0|S_{n-1}\notin C_5, S_{n-2}\notin C_4\}$,

$G_3=\{S_n\in C_1|S_{n-1}\in C_2\}$ $G_4=\{S_n\in C_1|S_{n-1}\in G_1\}$, $G_5=\{S_n\in C_1|S_{n-1}\in C_2\}$

$G_6=\{S_n\in C_1|S_{n-1}\notin(G_1\cup G_2\cup C_2)\}$, $G_7=\{S_n\in C_2\}$, $G_8=\{S_n\in C_3|S_{n-1}\in C_4\}$,

$G_9=\{S_n\in C_3|S_{n-1}\notin C_4\}$, $G_{10}=\{S_n\in C_4\}$, $G_{11}=\{S_n\in C_5\}$

Obviously, a stochastic process $\{S_n|n\geq0\}$ 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 of Ref.[8]). 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 (next page). From transition matrix P, for any two states $G_i,G_j\in\Gamma$, the state G_i is equivalent to G_j, so $\{S_n|n\geq0\}$ is irreducible, and for any G_j, it is

nonrecurrent. Therefore, the chain exists stationary distrubution$\{\pi_j, G_j \in \Gamma\}$,and $\lim_{m\to\infty}\left(\frac{1}{m}\sum_{n=1}^{m} p_{i,j}^{(n)}\right)=\pi_j$, where $p_{i,j}^{(n)}=P\{S_{(m+n)}\in G_j | S_m \in G_i\}$,$(G_i, G_j \in, m\geq 0, n\geq 1)$. From the equations below, which π_j (j=0,1,…,11) satisfies[8],

$$\boldsymbol{P}=\begin{bmatrix}
0 & 0 & \frac{1}{4} & 0 & \frac{3}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & \frac{5}{16} & 0 & 0 & \frac{11}{16} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{2}{3} & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{8} & 0 & \frac{1}{4} & 0 & \frac{3}{8}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{11}{44} & 0 & \frac{6}{44} & \frac{12}{44} & \frac{15}{44}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{6} & \frac{1}{3} & \frac{1}{4}\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

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

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

(9/130, 4/65, 1/5, 17/260, 3/65, 11/260, 3/20, 17/260, 4/65, 7/65, 4/65, 9/130).

Let its absorbing probabilities $p_j(n)=p\{S_n\in G_j\}$,j=0,1,…,11, and initial distribution probabilities $\{p_j\}=p\{S_0\in G_j\}$, j=0,1,…,11 of the chain, then the vector of $(u_{0,j},u_{1,j})$=(0,0) is the output by $G_{j,j}$=0,1,…,6. So AJHD is given by

$$\begin{aligned}\Sigma &= \sum_{j=7}^{11}\lim_{m\to\infty}\frac{1}{m}\sum_{n=1}^{m} p_j(n)=\sum_{j=7}^{11}\lim_{m\to\infty}\left(\frac{1}{m}\sum_{n=1}^{m}\sum_{G_i\in\Gamma} p_{i,j}^{(n)} p_i\right)\\
&=\sum_{j=7}^{11}\sum_{G_i} P_i\left(\lim_{m\to\infty}\left(\frac{1}{m}\sum_{n=1}^{m} p_{i,j}^{(n)}\right)\right)=\sum_{j=7}^{11}\lim_{m\to\infty}\left(\frac{1}{m}\sum_{n=1}^{m} p_{i,j}^{(n)}\right)\\
&=\sum_{j=7}^{11}\pi_j.\end{aligned}$$

Therefore Σ=19/52. The AJHD of τ-NJSF is 19/52.

4 Applications to ECC

The execution time of ECC schemes such as the ECDSA is 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 up by precomputing 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 Shamir Method by using JSF_3 yields a technique which requires approximately l doublings and 19l/52 general additions (on average). In other words, that sometimes works almost 9% faster than that by using the Joint Sparse Form.

The front type can also be sped up. The simplest approach is described below. Suppose that the order r of the private key space is less than 2^{2l}. Let $Q=2^lP$ then $k=a+b2^lQ$, thus compute $k=aP+bQ$, one applies Alg.1 to generate JSF_3 for integers a, b. This technique of computing it by using JSF_3 requires approximately $2l$ doublings and 19l/52, a saving of approximately 33l/52 general additions over the addition-subtraction method, and approximately 7l/52

over that by using the Joint Sparse Form. If the Elliptic Curves are particular curves, as Koblitz Curves, there may the form with width-3, analogous to JSF_3. So, it would be of interest to construct the form which can be applied to Koblitz Curves.

References:

[1] Cohen H. A Course in Computational Algebraic Number Theory. Berlin: Springer-Verlag, 1993.

[2] Gordon DM. A survey of fast exponentiation methods. Journal of Algorithms, 1998,27(1):129−146.

[3] Brown M, Hankerson D, Lopez J, Menezes A. Software implementation of NIST elliptic curves over prime fields. In: Naccache D, ed. Topics in Cryptology—CT-RSA 2001. LNCS 2020, Berlin: Springer-Verlag Heidelberg, 2001. 250−265.

[4] Avanzi R. On multi-exponentiation in cryptography. 2003. http://citeseer.ist.psu.edu/avanzi02multiexponentiation.html

[5] Bernstein DJ. Pippenger'2 exponentiation algorithm. 2002. http://cr.rp.to/papers.html

[6] Ciet M, Lange T, Sica F, Quisquater JJ. Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms. In: Biham E, ed. Advances in Cryptology-Eurocrypt 2003, LNCS 2656. Berlin, Heidelberg: Springer-Verlag, 2003. 388−400.

[7] Morain F, Olivos J. Speeding up the computations on an elliptic curve using addition-subtraction chains. Informatique Theorique et Applications/Theoritical Informatics and Applications, 1990,24(6):531−543.

[8] Chung KL. Elementary Probability Theory with Stochastic Processes. 3rd ed. Berlin: Springer-Verlag, 1978.

ZHANG Ya-Juan was born in 1974. She is a Ph.D. candidate at the Information Engineering University. Her current research area is information security.

KUANG Bai-Jie was born in 1976. He is a docent at the Department of Network Engineering, Information Engineering University. His current research area is information security.

ZHU Yue-Fei was born in 1962. He is a professor and doctoral supervisor at the Department of Network Engineering, Information Engineering University. His research area is information security.