Fast Elliptic Scalar Multiplication using New Double-base Chain and Point Halving

K.W. Wong[1*], Edward C.W. Lee[1], L.M. Cheng[1], and Xiaofeng Liao[2]

[1]Department of Electronic Engineering, City University of Hong Kong,
83 Tat Chee Avenue, Kowloon Tong, Hong Kong

[2]Department of Computer Science and Engineering, Chongqing University,
Chongqing, 400044, P.R. China

*Corresponding Author: K.W. Wong (e-mail: itkwwong@cityu.edu.hk)

Abstract

The fast implementation of elliptic curve cryptosystems relies on the efficient computation of scalar multiplication. Based on the double-base chain representation of scalar using powers of 2 and 3, we propose a new representation with powers of ½ and 3 instead. Thus the efficient point halving operation can be incorporated in the new double-base chain to achieve fast scalar multiplication. Experimental results show that our approach leads to a lower complexity which contributes to the efficient implementation of elliptic curve cryptosystems.

Keywords: Elliptic Curve Cryptography (ECC), Double-base Chain, Point Halving

1. Introduction

The popularity of private electronic communication and e-commerce stimulates the need to develop fast and secure cryptographic algorithms. Public key cryptography is a kind of cryptographic approach in which the encryption key can be made public to get rid of the key distribution problem that private key cryptography suffers. In particular, elliptic curve cryptography (ECC) is a public key cryptographic scheme with great potential. It was first proposed by Koblitz [1] and Miller [2] independently. In recent years, it is extensively studied

and implemented by mathematicians, cryptographers and computer scientists around the world because of its superior security over existing public key cryptographic algorithms. The best algorithm known for solving the underlying mathematical problem of ECC, referred to as the elliptic curve discrete logarithm problem (ECDLP), takes full exponential time. On the contrary, sub-exponential-time algorithms are known for tackling the integer factorization and the discrete logarithm problems that RSA and DSA is relied on, respectively [3]. This implies that the algorithms for solving the elliptic curve discrete logarithm problem become infeasible much more rapidly as the problem size increases than those algorithms for tackling the integer factorization and the discrete logarithm problems. For this reason, ECC offers a security level equivalent to RSA and DSA while using a far smaller key size.

ECC operates on the multiplicative group in finite field. A key factor for its fast implementation is how to compute the scalar multiplication kP efficiently, where k is a large integer and P is a point on the elliptic curve. Various fast algorithms have been proposed for this purpose. Traditionally, the integer k is represented in binary form and the double-and-add method is applied to calculate kP. This means that the accumulator is always doubled at each bit. If there's a "1" encountered, P is added to the accumulator. In order to reduce the amount of computation, the length of the representation of k and the number of non-zero elements in it should be kept as small as possible. The non-adjacent form (NAF), which is a signed representation, is proposed for this purpose [4]. This form guarantees that no two adjacent terms are both non-zero. In ECC, there is frequently a need to compute the value of $aP+bQ$ such as signature verification in Elliptic Curve Digital Signature Algorithm (ECDSA). In order to perform this computation efficiently, the joint sparse form (JSF) was proposed [5]. It is based on NAF and is an optimal signed binary representation of a pair of integers that leads to more double zero positions. The use of an efficient endomorphism can increase the computational speed of the scalar computation if the elliptic curve admits the endomorphism. In [6], anomalous binary curves on which the Frobenius map can be applied were proposed by

Koblitz. For this type of curves, an efficient scalar representation, referred to as reduced τ-adic non-adjacent form (RTNAF), was suggested by Solinas [7]. Under this RTNAF representation, τP is performed by squaring the x and y coordinates of point P. If normal basis is adopted, this operation can be done in a very short time as a squaring is equivalent to a shift operation.

Recently, the classical approach of representing the scalar in binary form and then performing the scalar multiplication by a standard double-and-add method has been advanced. In particular, there are suggestions that the scalar k can be expressed in a mixed base. In [8], a ternary / binary approach making use of the efficient triple ($3P$) and double ($2P$) of point P was proposed for fast scalar multiplication. A similar idea was suggested in [9] that the scalar is represented as a double-base (bases 2 & 3) chain. On the other hand, point halving was proposed independently by Knuden [10] and Schroeppel [11]. They suggested that point doubling in the double-and-add method can be replaced by a faster point halving operation. A detail analysis of the speed advantage of employing point halving instead of point doubling can be found in [12]. Moreover, point halving can be combined with Frobenius endomorphism so as to speed up the corresponding operation in Koblitz curve by 25% [13,14].

In this paper, we propose a new double-base chain representation with bases ½ and 3 for the incorporation of point halving in scalar multiplication. Experimental results show that our approach leads to a lower complexity in computing scalar multiplication. The organization of the remaining parts of this paper is as follows. In Section 2, the double-base chain representation and the idea of point halving will be described in detail. Then we propose, in Section 3, our new double-base chain representation and the utilization of point halving for scalar multiplication. In Section 4, experimental results of the computation complexity of our approach will be presented. Conclusions will be drawn in the last section.

2. Background Theory

Our approach is based on two existing techniques. The double-base chain representation of a scalar will first be introduced in Section 2.1 while and the operations of point halving will be described in Section 2.2.

2.1 Double-base Chain Representation

Ciet *et al* [8] have proposed a ternary / binary approach for fast ECC scalar multiplication. It makes use of the efficient doubling ($2P$), tripling ($3P$), and quadrupling ($4P$) of a point P. A similar idea was suggested in [9] that an integer k is represented in double-base number system as the sum or difference of mixed powers of two and three, as given by Eq. (1)

$$k = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i} \qquad \text{with} \quad s_i \in \{1, -1\} \quad \text{and} \quad b_i, t_i \geq 0 \tag{1}$$

If the sequences of binary and ternary exponents decrease monotonically, i.e., $b_1 \geq b_2 \geq ... \geq b_m \geq 0$ and $t_1 \geq t_2 \geq ... \geq t_m \geq 0$, a double-base chain is formed. As a result, fast computation of scalar multiplication is achieved by the following recursive calculations.

$$K_1 = 1, \qquad K_i = 2^u 3^v K_{i-1} + s_i \qquad \text{with} \quad i \geq 2, \quad s_i \in \{1, -1\} \tag{2}$$

where u is the difference of two consecutive binary exponents, and v is the difference of two consecutive ternary exponents.

Take the example of 314159 as used in [9]. Its double-base chain representation is

$$314159 = 2^{12}3^4 - 2^{11}3^2 + 2^8 3^1 + 2^4 3^1 - 2^0 3^0 \tag{3}$$

The calculation successively computes $17P$, $409P$, $6545P$ and finally $314159P$, as illustrated in Table 1.

i	K_i	s	u	v
1	1	1	0	0
2	$18\,K_1 - 1 = 17$	-1	1	2

3	$24\,K_2 + 1 = 409$	1	3	1
4	$16\,K_3 + 1 = 6545$	1	4	0
5	$48\,K_4 - 1 = 314159$	-1	4	1

Table 1: Procedures in calculating 314159P in five iterations.

If prime field is chosen, it needs to calculate 13 inversions, 55 squarings and 95 multiplications. The analyses made in [9] show that this method is faster than the classical binary, 2-NAF, 4-NAF and the ternary / binary approach proposed in [8], over both binary and prime fields. Moreover, it does not require any precomputation.

2.2 Point Halving Operations

Point halving was proposed independently by Knuden [10] and Schroeppel [11]. It is the reverse operation of point doubling. Let $P = (x,y)$ be a point on the elliptic curve defined over binary field using affine coordinates. A point doubling requires to calculate the coordinates of the point $Q = 2P = (u,v)$ using the following equations:

$$\lambda = x + y/x \tag{4}$$

$$u = \lambda^2 + \lambda + a \tag{5}$$

$$v = x^2 + u\,(\lambda + 1) \tag{6}$$

Point halving is just the opposite, *i.e.,* given $Q = (u,v)$, find $P = (x,y)$ such that $Q = 2P$. It is computed by solving Eq. (5) for λ, Eq. (6) for x, and finally, Eq. (4) for y. This means that we have to solve $\lambda^2+\lambda = u + a$ for λ, $x^2 = v + u\,(\lambda +1)$ for x, and finally obtain $y = \lambda\, x + x^2$.

If all the point doublings required in the traditional double-and-add method are replaced by the faster point halving operation, the computation speed could be up to 39% [10] and 50% [11] faster. A detail analysis of the computational complexity of point halving was made in [12]. It was reported that the point halving method is 15% to 24% faster than point doubling.

Moreover, this approach performs better when the point P is not known in advance and the inversion-to-multiplication ratio is small.

3. The Proposed Approach

For fast ECC scalar multiplication, we propose a new double-base chain representation for scalars. Instead of mixed powers of 2 and 3, our idea is to represent the scalar by a new double-base chain with monotonic decreasing powers of ½ and 3. As a result, point doubling and quadrupling can be replaced by point halving while keeping the tripling operations. The algorithm in finding such a new double-base chain is described as follows.

First of all, we multiply k with a large power of 2, say, 2^q. From the experimental results, we choose 2^q to be a value around the field size. Then we find the remainder k' after modulo the field size p, as given by Eq. (7).

$$k' = 2^q k \bmod p \tag{7}$$

The next step is to obtain the double-base chain of k' with powers of 2 and 3 in the form of increasing binary exponents but decreasing ternary exponents. We achieve this by an iterative approach. First, we find n such that $k = 0 \bmod n$, with the trial of n in the order of 6, 4, 3 and 2, respectively. If $k = 0 \bmod 6$, we return $2 \times 3\left(\frac{k}{2 \times 3}\right)$. If $k = 0 \bmod 4$, we return $2 \times 2\left(\frac{k}{2 \times 2}\right)$. If $k = 0 \bmod 3$, we return $3(k/3)$. If $k = 0 \bmod 2$, we return $2(k/2)$. If there is no suitable match after all the four trials, we find k_2 - a power of 2 that is the closest to k, and return the absolute value of the difference, i.e., $|k - k_2|$. We choose a power of 2 as an approximation to k because doubling (which becomes halving later) is more favorable than tripling. As the returned value $|k - k_2|$ is getting smaller and smaller, it can always be approximated by a lower power of 2 in next approximation. As a result, there will not be adjacent terms in the double-base chain having the same binary exponents. Thus, the binary

exponents in this double-base chain keep strictly decreasing and so triple-and-add operation is not required in the scalar multiplication. Moreover, the sequence of ternary exponents in this double-base chain will only increase or remain unchanged, but will never decrease. This is because whenever $k = 0$ or 3 mod 6, the ternary exponent will increase. For the rest of values of k, the ternary exponents remain unchanged. In the worse case, we can use purely a sequence of power of 2 while keeping all ternary exponents zero to represent the scalar k. The recursion stops until k is equal to 1, a power of 2 or a power of 3, i.e., a positive number that can be represented by 2^b3^t for any non-negative integers b and t.

This iterative algorithm will return the terms in an order from the highest power of 2 times the lowest power of 3 to the lowest power of 2 times the highest power of 3. If we reverse the order of the terms, i.e., the last term becomes the first term, the expression becomes the desired double-base chain with increasing binary exponents but decreasing ternary exponents. Finally, we divide the double-base chain by 2^q to make all the binary exponents negative but with decreasing magnitude. The ternary exponents are unaffected and are all positive or zero with decreasing magnitude. This is actually a new double-base chain with decreasing powers of ½ and 3, with value equal to k. To summarize, the representation of k is given by Eq. (8).

$$k = \frac{k'}{2^q} = \frac{\sum_{i=1}^{m} s_i 2^{b_i'} 3^{t_i}}{2^q} = \sum_{i=1}^{m} s_i \left(\frac{1}{2}\right)^{(q-b_i')} 3^{t_i} \quad \bmod p \tag{8}$$

where $s_i \in \{1, -1\}$, $0 \le b'_1 < b'_2 < ... < b'_m$, $t_1 \ge t_2 \ge ... \ge t_m \ge 0$, $q \ge b_i' \ \forall i$.

For the example used in Section 2.1, if the field size p is chosen as 314161, the scalar 314159 becomes 52017 after multiplied 2^{17} and mod 314161. The steps in finding the new double-base chain representation of 314159 are shown in Table 2.

Step	*Divisible by*	*Return*
0	/	52017
1	3	3 (17339)
2	/	3 (2^{14} + 955)
3	/	3 (2^{14} + (2^{10} – 69))
4	3	3 (2^{14} + (2^{10} – 3(23)))
5	/	3 (2^{14} + (2^{10} – 3(2^4 + 7)))
6	/	3 (2^{14} + (2^{10} – 3(2^4 + (2^3 – 1))))
Double-base chain		$2^{14}3^1 + 2^{10}3^1 – 2^43^2 – 2^33^2 + 2^03^2$
After reversed the terms		$2^03^2 – 2^33^2 – 2^43^2 + 2^{10}3^1 + 2^{14}3^1$
After divided by 2^{17}		$(½)^{17}3^2 – (½)^{14}3^2 – (½)^{13}3^2 + (½)^73^1 + (½)^33^1$

Table 2: Steps in obtaining a new double-base chain for k = 314159.

More examples with k multiplied by different values of q over the prime field with size p = 314161 are given in Table 3. When q = 15 and 19, there are five terms in the new double-base chain, same as that in the original chain. However, when q = 24, there are only four terms, i.e., one term fewer although the highest power of ½ and 3 are both larger. Besides, the new double-base chain obtained when multiplying by 2^{17} is exactly equal to that by 2^{19} although the values of q and hence k' are different.

q	$k' = 2^q k \bmod p$	New double-base chain for k = 314159
15	248625	$(½)^{15}3^4 – (½)^{10}3^4 – (½)^73^3 – (½)^33^2 + (½)^03^2$
17	52017	$(½)^{17}3^2 – (½)^{14}3^2 – (½)^{13}3^2 + (½)^73^1 + (½)^33^1$
19	208068	$(½)^{17}3^2 – (½)^{14}3^2 – (½)^{13}3^2 + (½)^73^1 + (½)^33^1$
24	60795	$(½)^{24}3^5 + (½)^{21}3^4 – (½)^{15}3^3 + (½)^{11}3^2$

Table 3: Some double-base chain representations for k = 314159.

4. Experimental Results

We perform our experiments on a Pentium D 3.00GHz platform using C++ with the MIRACL library version 5.0 which consists of a collection of routines for treating large integers. The library covers a full set of functions for addition, subtraction, multiplication, division and modulo operations of large integers.

Tables 4 to 6 show the results for binary field with field size ranged from 163-bit to 283-bit while Table 7 lists the data for prime field with field size 192-bit, both use NIST recommended fields. For each field, we generate five different values of k randomly. Some of them have length equal to the field size while some are shorter. The original and new double-base chains for each generated k are found. The variable b_{max} is the largest binary exponent while t_{max} is the largest ternary exponent in the original double-base chain. In our proposed method, the largest binary exponent b'_{max} is counted before dividing the double-base chain by 2^q. After obtained the chains, we calculate the corresponding number of curve operations required. For the original double-base chain, point doubling (D), double-and-add (DA), tripling (T), triple-and-add (TA), quadrupling (Q), and quadruple-and-add (QA) operations are used [9]. However, in our proposed method, only point halving (H), half-and-add (HA) and tripling (T) operations are required. Note that the sum of the number of halving (H) and halving-and-add (HA) operations equals to q while the number of tripling (T) equals to t_{max}. The number of terms in the two types of double-base chain is also listed in the tables. Sometimes our approach leads to fewer terms but sometimes more. It seems that there is not a consistent trend.

For binary fields, on counting the number of curve operations needed for the original method using doublings, we prefer $2(2P)$ to $4P$ as two consecutive doubling operations are faster than one quadrupling. However, we prefer a single quadruple-and-add operation rather than a doubling followed by a double-and-add because the former requires one multiplication fewer than the latter regardless of the number of squarings.

To compare the complexity of the two methods, the equivalent numbers of inversions [i], squarings [s] and multiplications [m] for binary field are calculated based on Table 8 [9,12]. The results are also included in Tables 4 to 6. They indicate that our approach usually requires only about half the number of inversions, one-third the number of squarings, and also a slightly fewer number of multiplications. This shows that the computational complexity for scalar multiplication is reduced substantially. However, as point halving has been studied in binary field only [10,12], no complexity data for calculating a point halving in prime field are available. Therefore we cannot compare the complexity of the two methods directly for prime field.

Table 9 lists the time needed for converting k to k', the time required to find the increasing binary but decreasing ternary exponents double-base chain, the time for dividing the chain by 2^q and finally the total conversion time. The data indicate that the time to find the double-base chain representation for k' dominates the total conversion time while the additional time required in the pre- and post-processing is comparatively insignificant. In particular, the time required for dividing the chain by 2^q is very fast because we don't need to perform an actual division, but just need to subtract the binary exponents from q.

Method using original double-base chain											
Bit length of k	b_{max}	t_{max}	**No. of terms**	**D**	**DA**	**T**	**TA**	**QA**	**[i]**	**[s]**	**[m]**
150-bit	118	20	40	58	14	18	2	23	140	302	616
158-bit	115	27	42	58	19	24	3	19	145	315	672
163-bit$_1$	82	51	37	38	16	45	6	14	139	352	729
163-bit$_2$	118	28	47	56	24	25	3	19	149	327	720
163-bit$_3$	66	61	52	36	14	32	29	8	156	327	763
Our proposed method											
Bit length of k	**q**	**b'_{max}**	**t_{max}**	**No. of terms**	**H**	**HA**	**T**	**-**	**-**	**[i]**	**[s]**	**[m]**
150-bit	165	163	21	41	125	40	21	-	-	61	84	597
158-bit	160	160	27	43	118	42	27	-	-	69	108	635
163-bit$_1$	163	161	26	35	129	34	26	-	-	60	104	610
163-bit$_2$	163	160	23	36	128	35	23	-	-	58	92	592
163-bit$_3$	163	159	22	43	121	42	22	-	-	64	88	606

Table 4: Number of curve and field operations for NIST B-163 binary field.

Method using original double-base chain											
Bit length of k	b_{max}	t_{max}	**No. of terms**	**D**	**DA**	**T**	**TA**	**QA**	**[i]**	**[s]**	**[m]**
189-bit	74	72	52	38	14	46	26	11	172	394	868
192-bit	169	14	66	76	31	11	3	31	186	377	845
224-bit	163	38	61	86	27	30	8	25	209	434	947
233-bit$_1$	172	38	58	88	26	36	2	29	212	464	970
233-bit$_2$	164	43	52	94	28	41	2	21	209	446	955
Our proposed method											
Bit length of k	**q**	**b'_{max}**	**t_{max}**	**No. of terms**	**H**	**HA**	**T**	**-**	**-**	**[i]**	**[s]**	**[m]**
189-bit	238	232	37	55	184	54	37	-	-	91	148	897
192-bit	232	228	39	54	179	53	39	-	-	92	156	896
224-bit	234	232	40	54	181	53	40	-	-	93	160	907
233-bit$_1$	239	223	40	48	192	47	40	-	-	87	160	899
233-bit$_2$	232	228	39	56	177	55	39	-	-	94	156	902

Table 5: Number of curve and field operations for NIST B-233 binary field.

Method using original double-base chain											
Bit length of k	b_{max}	t_{max}	**No. of terms**	**D**	**DA**	**T**	**TA**	**QA**	**[i]**	**[s]**	**[m]**
256-bit$_1$	114	89	58	70	20	64	25	12	228	513	1113
256-bit$_2$	211	28	76	100	31	24	4	40	243	510	1083
283-bit$_1$	156	80	57	82	24	73	7	25	243	593	1204
283-bit$_2$	202	51	74	106	38	45	6	29	259	554	1213
283-bit$_3$	256	17	93	126	48	14	3	41	276	533	1219
Our proposed method											
Bit length of k	q	b'_{max}	t_{max}	**No. of terms**	**H**	**HA**	**T**	**-**	**-**	**[i]**	**[s]**	**[m]**
256-bit$_1$	278	278	42	68	211	67	42	-	-	109	168	1051
256-bit$_2$	286	283	47	71	216	70	47	-	-	117	188	1111
283-bit$_1$	279	278	40	73	207	72	40	-	-	112	160	1054
283-bit$_2$	284	280	50	68	217	67	50	-	-	117	200	1119
283-bit$_3$	281	279	44	70	212	69	44	-	-	113	176	1077

Table 6: Number of curve and field operations for NIST B-283 binary field.

Method using original double-base chain											
Bit length of k	b_{max}	t_{max}	**No. of terms**	**DA**	**T**	**TA**	**Q**	**QA**	**[i]**	**[s]**	**[m]**
150-bit	118	20	40	14	18	2	29	23	111	461	784
158-bit	115	27	42	19	24	3	29	19	116	483	836
163-bit	82	51	37	16	45	6	19	14	120	463	838
189-bit	74	72	52	14	46	26	19	11	153	531	974
192-bit	169	14	66	31	11	3	38	31	148	584	1066
Our proposed method											
Bit length of k	q	b'_{max}	t_{max}	**No. of terms**	**H**	**HA**	**T**	**-**	**-**	**[i]**	**[s]**	**[m]**
150-bit	191	190	32	47	145	46	32	-	-	-	-	-
158-bit	190	186	34	48	143	47	34	-	-	-	-	-
163-bit	193	190	32	47	147	46	32	-	-	-	-	-
189-bit	191	190	33	43	149	42	33	-	-	-	-	-
192-bit	192	191	25	54	139	53	25	-	-	-	-	-

Table 7: Number of curve and field operations for NIST P-192 prime field.

Curve operation	Complexity
Halving (H)	2[m]
Half-and-add (HA)	1[i] + 5[m]
Tripling (T)	1[i] + 4[s] + 7[m]

Table 8: The complexity for different curve operations over binary field.

Bit length of k	**Conversion of k**	**Time to convert k to k' / ms**	**Time to convert k' to a double-base chain / ms**	**Time to divide the chain by 2^q / ms**	**Total conversion time / ms**
B-163: Binary field size 163-bit with irreducible polynomial $2^{163}+2^7+2^6+2^3+1$					
150-bit	$2^{165}k \bmod (\mathrm{B}-163)$	0.0151	3.8551	0.0019	3.8721
158-bit	$2^{160}k \bmod (\mathrm{B}-163)$	0.0155	4.0636	0.0020	4.0811
163-bit$_1$	$2^{163}k \bmod (\mathrm{B}-163)$	0.0150	3.3644	0.0017	3.3811
163-bit$_2$	$2^{163}k \bmod (\mathrm{B}-163)$	0.0143	3.4256	0.0017	3.4416
163-bit$_3$	$2^{163}k \bmod (\mathrm{B}-163)$	0.0150	4.0270	0.0019	4.0439
B-233: Binary field size 233-bit with irreducible polynomial $2^{233}+2^{74}+1$					
189-bit	$2^{238}k \bmod (\mathrm{B}-233)$	0.0179	5.3730	0.0025	5.3934
192-bit	$2^{232}k \bmod (\mathrm{B}-233)$	0.0168	5.3333	0.0023	5.3524
224-bit	$2^{234}k \bmod (\mathrm{B}-233)$	0.0189	5.3164	0.0023	5.3376
233-bit$_1$	$2^{239}k \bmod (\mathrm{B}-233)$	0.0183	4.8176	0.0022	4.8381
233-bit$_2$	$2^{232}k \bmod (\mathrm{B}-233)$	0.0183	5.5036	0.0025	5.5244
B-283: Binary field size 283-bit with irreducible polynomial $2^{283}+2^{12}+2^7+2^5+1$					
256-bit$_1$	$2^{278}k \bmod (\mathrm{B}-283)$	0.0211	6.7055	0.0029	6.7295
256-bit$_2$	$2^{286}k \bmod (\mathrm{B}-283)$	0.0211	7.0172	0.0031	7.0414
283-bit$_1$	$2^{279}k \bmod (\mathrm{B}-283)$	0.0215	7.1392	0.0029	7.1636
283-bit$_2$	$2^{284}k \bmod (\mathrm{B}-283)$	0.0223	6.8095	0.0029	6.8347
283-bit$_3$	$2^{281}k \bmod (\mathrm{B}-283)$	0.0213	6.9416	0.0029	6.9658

Table 9: The time for each conversion step and the total conversion time.

5. Conclusions

We have proposed a new double-base chain representation for a scalar k with mixed powers of ½ and 3. The advantage of this representation is that all point doublings required in the original chain can be replaced by the faster point halving operations. Experimental results show that for binary fields, our approach requires only about half the number of inversions, one-third the number of squarings, and a slightly fewer number of multiplications when compared with the scalar multiplication using the original double-base chain. The substantially reduced computational complexity contributes to the efficient implementation of elliptic curve cryptosystems.

Acknowledgement

The work presented in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. 9041035 (CityU 121305)].

References

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, 48, pp. 203–209, 1987.

[2] V.S. Miller, "Use of elliptic curves in cryptography," in H.C. Williams Ed., *Advances in Cryptology – CRYPTO'85, LNCS* 218, Springer-Verlag, pp. 417–426, 1986.

[3] S.A. Vanstone, "Next generation security for wireless: elliptic curve cryptography", *Computers and Security*, vol. 22, no. 5, pp. 412-415, 2003.

[4] F. Morain & J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains," Information Theory Applications, vol. 24, pp. 531-543, 1990.

[5] J.A. Solinas, "Low-Weight Binary Representations for Pairs of Integers," *Technical Report CORR 2001-41*, CACR, available at: *www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-41.ps*.

[6] N. Koblitz, "CM curves with good cryptographic properties," *Advances in Cryptology – Crypto'91*, LNCS 547, Springer-Verlag, pp. 279-287, 1992.

[7] J. Solinas, "Efficient arithmetic on Koblitz curves," *Designs, Codes, and Cryptography* 19, pp. 195-249, 2000.

[8] M. Ciet, M. Joye, K. Lauter, and P.L. Montgomery, "Trading Inversions for Multiplications in Elliptic Curve Cryptography," *Cryptology ePrint Archive, Report 2003/257, 2003*. Also to appear in *Design, Codes and Cryptography*.

[9] V.S. Dimitrov, L. Imbert, and P.K. Mishra, "Fast Elliptic Curve Point Multiplication using Double-Base Chains, *Cryptology ePrint Archive, Report 2005/069*, 2005.

[10] E.W. Knudsen, "Elliptic Scalar Multiplication using Point Halving," *ASIACRYPT'99, LNCS 1716*, K.Y. Lam, E. Okamoto and C. Xing Ed., pp. 135 – 149, 1999.

[11] R. Schroeppel, "Elliptic Curve Point Ambiguity Resolution Apparatus and Method," *International Patent Application Number PCT/US00/31014*, filed 9 November 2000.

[12] K. Fong, D. Hankerson, J. Lopez, and A. Menezes, "Field Inversion and Point Halving Revisited," *IEEE Transactions on Computers*, vol. 53, no. 8, pp. 1047 – 1059, 2004.

[13] R.M. Avanzi, C. Heuberger, and H. Prodinger, "Minimality of the Hamming Weight of the τ-NAF for Koblitz Curves and Improved Combination with Point Halving", *Cryptology ePrint Archive, Report 2005/225*, 2005.

[14] R.M. Avanzi, M. Ciet, F. Sica, "Faster Scalar Multiplication on Koblitz Curves Combining Point Halving with the Frobenius Endomorphism" *Public Key Cryptography* (*PKC 2004*), *LNCS 2947,* F. Bao, R. H. Deng, J. Zhou Eds., , pp. 28-40. Springer-Verlag, 2004.