An Improved Scalar Multiplication Over $GF(2^m)$ for ECC

S. Revathi
Research scholar, Mathematics,
Theivanai Ammal College for Women,
villupuram-605 602, T.N India,

A. R. Rishivarman
Assistant Professor, Mathematics,
Theivanai Ammal College for women,
villupuram-605 602, T.N India,

ABSTRACT
Since the introduction of public-key cryptography by Diffe and Hellman in 1976, the potential for the use of the discrete logarithm problem in public-key cryptosystems has been recognized. Although the discrete logarithm problem as first employed by Diffe and Hellman was defined explicitly as the problem of finding logarithms with respect to a generator in the multiplicative group of the integers module a prime, this idea can be extended to arbitrary groups and in particular, to elliptic curve groups. The resulting public – key systems provide relatively small block size, high speed, and high security. This paper identified an efficient performance of concurrent algorithm using complementary recoding over $GF(2^m)$ for scalar multiplication in the polynomial basis (PB) to use in an elliptic curve cryptosystems, which enhances security. So this scheme is of less computation cost which is valuable in applications with limited memory, communications bandwidth or computing power.

Keywords
Secret sharing, Elliptic Curve Cryptography (ECC), $GF(2^m)$, Irreducible polynomial, ECDLP.

1. INTRODUCTION
There are three families of public–key algorithms that have considerable significance in current data security practice. They are integer factorization, discrete logarithm and elliptic curve based schemes [1, 2]. Integer factorization based schemes such as RSA [3] and discrete logarithm based schemes such as Diffe -Hellman [4] provide intuitive ways of implementation. However both methods admit of sub-exponential algorithm of cryptanalysis [5]. In this regard elliptic curve cryptography, first introduced Koblitz [1] and Miller [2] may be the most cryptographic method available [6, 7]. The best current brute force algorithm for cryptanalysis of ECC require $O(n^{1/2})$ steps where n is the order of the additive group. For example, using the best current brute force algorithms ECC with a key size of 173 bits provides the same level of cryptographic security as RSA with a key size of 1024 bits. This results in smaller system parameters band width savings, faster implementations and lower power consumptions. In addition, elliptic curve over finite fields offer an inexhaustible supply finite abelian groups, thus allowing more flexible fields selections than conventional discrete logarithm schemes [8]. Because of these advantages ECC has attracted extensive attention in recent years [9, 10]. This paper identified an efficient performance of concurrent algorithm using complementary recoding over $GF(2^m)$ for scalar multiplication in the polynomial basis (PB) $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4\}$, where α is the root of an irreducible polynomial $f(x) = x^5 + x^2 + 1$ (1) to use in ECC.

In the elliptic curve cryptosystem, main operations such as key agreement, signature generation, signing and verification involve scalar multiplication. The speed of scalar multiplication plays an important role in the efficiency of whole system. Fast multiplication is very essential in some environments such as constrained devices, central servers, where large number of key agreements, signature generations and verification occurs.

In this paper, scalar multiplication using concurrent algorithm is proposed with the help of complementary recoding. This is achieved by representing the integer

$$k = \sum_{i=0}^{l-1} k_i 2^i = (100..0)_{(l+1)bits} - \bar{k} - 1. \quad (2)$$

This computation is very simple and efficient when compared to other standard methods. Earlier in [11] concurrent algorithm has been proposed, but employed only the binary coding. We employed as a first time Complementary recoding to speedup the arithmetic operation in the concurrent algorithm [18] and reduced the computation time considerably [19, 20].

2. $GF(2^5)$ ARITHMETIC OPERATIONS
There is a representations for an element of a finite field $GF(2^5)$, The polynomial basis (PB), where $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4\}$ where α is a root of an irreducible polynomial $f(x) = x^5 + x^2 + 1$ over the field F_2. An element in $GF(2^5)$ can then be represented as a polynomial $\{c_0 + c_1\alpha + c_2\alpha^2 + c_3\alpha^3 + c_4\alpha^4 \ / c_i = 0$ or $1\}$ or in vector form $[c_o, c_1, c_2, c_3, c_4]$

Hence it is easy to see the elements of $GF(2^5)$ and $[Res_{F_{2^5}/F_2} GF(2^5)]$ (F_2). Observe that there is a isomorphism of sets.

$$GF(2^5) \cong [Res_{F_{2^5}/F_2} GF(2^5)]\ (F_2) \quad (3)$$

The efficiency of elliptic curve algorithm heavily depends on the performance of the under lying field arithmetic operations. These operations include addition, Subtraction, Multiplication, and Inversion.

3. SCALAR MULTIPLICATION
The elliptic curve cryptographic scheme requires the scalar multiplication defined as follows [12].

$$Q = kP = P + P + \cdots + P \ , \ k \text{ times} \quad (4)$$

where P denotes a point on the elliptic curve and k is a random integer with $1 \leq k \leq$ order $(P) \approx 2^m - 1$. 1. Algorithm gives the well-known double and add algorithm, also referred to as the binary method, to compute kP assuming that Q is initialized as an infinite point denoted by the symbol O.

In this paper the scalar multiplication is computed with concurrent algorithm which uses complementary recoding for reducing of hamming distances of the scalars and generated a cyclic group. It is considered a super singular elliptic curve $y^2 + y = x^3$ defined over $GF(2^5)$, the primitive polynomial chosen for constructing the finite field be $x^5 + x^2 + 1$.

Algorithm 1: Double –and-add -binary method
Input**:** k and P

Output**:** $Q = kP$

$/*$ convert the integer k into the binary representation $*/$

$Q = kP; k = (k_{t-1}, k_{t-2}, \ldots., k_0);\ k_i \in \{0,1\}$

1. $Q = O$;
2. for i from $t - 1$ down to 0 do
3. $Q = Q = Q$;
4. if $k_i = 1$, then $Q = Q + P$;
5. return Q:

As seen from algorithm 1, the expected number of point additions is approximately $0.5t$ and the number of doubles is exactly t. The expected number of point additions is actually equivalent to the average number of nonzero coefficients of k. To reduce the nonzero coefficients of k, the non-adjacent form (NAF) and modified Booth's encoding are commonly applied to achieve an expected value of $t/3$ and $3t/8$, respectively. The NAF representation of k means that all its nonzero digits, e.g. 1 of -1, are never adjacent to each other.

A detailed inspection shows that converting an operand into the corresponding NAF starts with the least significant bit (LSB), which is the counter direction while performing the double-and –add algorithm. Therefore, NAF may not be an appropriate choice to be embedded in the double-and-add algorithm, even though the NAF exhibits better encoding efficiency than the modified Booth's encoding. To overcome this deficiency, Satoh and Takano introduced complementary recoding to transfer the binary number k from the most significant bit (MSB). This method is very suitable for VLSI design.

The cost of multiplication depends on the length of the binary representation of 'k' and the number of 1's in this representation. If the representation $(k_{n-1} \ldots k_1\, k_0)_2$ has $k_{n-1} \neq 0$ then the number of doubling operation is $(n-1)$ and the number of addition operation is one less than the number of non-zero digits in $(k_{n-1} \ldots k_1\, k_0)_2$ .the number of non-zero digit is called the hamming weight of scalar multiplication. In an average, binary method requires $(n-1)$ doublings and $(n-1)/2$ additions.

For example, the integer $k = 729$ and the binary representation is $(1011011001)_2$, computation of $729P$ requires 9 doubles and 5 additions. Whenever the bits is 1, two elliptic curve operation such as ECDBL and ECADD will be made and if it is 0 , only one operation ,ECDBL required. So if we reduce number of ones in the scalar representation or hamming weight, we could speed up the above computation.

3.1 Addition –Subtraction Method

In 1951, Booth [9] proposed a new scalar representation called signed binary method and later Reitweisner [13] proved that every integer could be uniquely represented in this way. The property of this representation is that, of any consecutive digits, at most one is non-zero. Here the integer 'k' is represented as $\sum_{j=0}^{l-1} k_j 2^j$, where each $k_j \in \{-1,0,1\}$. Reitweisner's canonical representations have become to be called non- adjacent form (NAF) [14]. Fortunately, the NAF of 'k' is at most one digit longer than the {0,1}-radix 2-representstion. Algorithm 2 is for the conversion of an integer k into the NAF of the same using three digits $\{0,1,-1\}$-radix 2 representation and this conversion will take place from right-to-left.

Algorithm 2: Computation of NAF of an Integer
Input: positive integer k

Output: s(NAF representation of k)

$C = k;\ l = 0$

While $(c > 0)$

If (c is odd)

$s[l] = 2 - (c \bmod 4)$

$c = c - s[l]$

else
$s[l] = 0$

endif

$c = c/2;\ l = l + 1$

end while

Return s

The average hamming weight of signed binary representation is $n/3$ and it has the lower hamming weight than the binary representation. For example, the binary representation of 2927 is $(101101101111)_2$, the hamming weight is 9 and NAF of 2927 is $(01100\text{-}100\text{-}1000\text{-}1)_2$, the hamming weight is only 5. Here the hamming weight is reduced from 9 to 5, which improve the speed of the scalar multiplication .

One notable property of elliptic curve group is that the inverse of a point can be computed virtually free. This is the reason why a signed representation of the scalar is meaningful. The binary method is revised accordingly the new algorithm is called addition-subtraction method [14] given in the following algorithm.

Algorithm 3: addition and subtraction method
Input: k and P

Output: $Q = kP$

$s = \text{NAF}(k)$ $/*$ the NAF form of k stored in s $*/$

Q=0

For $i = n - 1$ down to 0

$Q = 2Q$

If $(s_I = 1)$

$Q = Q + P$

Endif $(s_I = -1)$

$Q = Q - P$

end If

Return Q

The algorithm performs $(n-1)$ doublings and $(n-1)/3$ additions in an average. In the case of elliptic curve scalar multiplication, the left-to-right evaluation stage is natural choice. The disadvantage of the addition and subtraction method is that is necessary to complete the recoding and store them before starting left-to-right evaluation stage . Hence it requires additional n-bit memory for the right-to left exponent recoding.

3.2 Mutual Opposite Form (MOF)

The left-to-right recoding method eliminates the need for recoding and storing the multiplier in advance[15]. Joye and Yen proposed first left –to-right recoding algorithm in 2000. In CRYPTO 2004, Okeya proposed a new efficient left-to-right recoding scheme called mutual opposite form (MOF) and the property of MOF is that

- Sign of adjacent non-zero bits (without considering 0 bits) are opposite
- Most non-zero bit and the least non-zero bit are 1 and -1, respectively
- All the positive integers can be represented by unique MOF

The n-bit binary string k is converted into a signed binary string by computing $mk = 2k - k$, where '$-$' stands for a bit wise subtraction. Algorithm given below is a simple and flexible conversion from n-bit binary string k to (k+1) bit MOF of the same.

Algorithm 4: left-to-right generation from binary to MOF

Input: n-bit binary string

$$d = d_{n-1}|d_{n-2}|\ldots.|d_1|d_0$$

Output: MOF of $d(md_n|\ldots.|md_1|md_0)$

$md_n = d_{n-1}$

for $i = n-1$ down to 1 do

$md_i = d_{i-1} - d_i$

$md_0 = -d_0$

return $(md_n, \ldots\ldots, md_1, md_0))$

The above algorithm converts the binary string to MOF from the most significant bit efficiently. The conversion of MOF representation of an integer.

Is highly flexible because, conversion can be made either from right- to- left or left-to-right. The output of MOF is same as the output of NAF.

4. COMPLEMENTARY RECODING

In 2003, chang et al. [16, 17] proposed an efficient method to compute the general multiplication by performing complement operation. Assume the binary representation of a scalar k is $(k_{n-1} \ldots k_1 k_0)_2$,the procedure for converting binary string using complementary method is given below.

$k = \sum_{i=1}^{l-1} k_i\, 2^i = (100\ldots0)_{(l+1)bits} - \bar{k} - 1$

where $\bar{k} = \bar{k}_{i-1}\bar{k}_{i-2}\ldots\bar{k}_0$,

And $\bar{k}_i = 0$ if $k_i = 1$

$\bar{k}_i = 1$ if $k_i = 0$ for $i = 0,1,2\ldots k-1$

Illustration

$k = 687 = (1010101111)_2$

$= (10000000000) - \bar{k} - 1$

$= (100000000)_2 - (0101010000)_2 - 1$

$= (10 - 10 - 10 - 1000 - 1)_2$

$= 1024 - 256 - 64 - 16 - 1 = 687$

The hamming weight of binary representation is 7 and signed binary representation using complementary recoding is just 5, here two elliptic curve addition operations have been saved. One addition operation requires $2S + 2M + I$ and this representation saves totally $4S + 4M + 2I$ operations.

4.1 The Proposed Algorithm

According to the data dependency of the three operations in point addition / doubling, there exists idle time for the consecutive point addition or doubling operation. An efficient way would be to increase the hardware utilization by introducing another sequence of point addition or doubling operations into the idle time. To accomplish this goal, we employ the interleaving scheme by dividing the scalar k into two parts, the high-order part $k_H = (k_{t-1}\ldots k_{(t-1/2)+1}, k_{(t-1)/2})$ and the low-order part $k_L = (k_{(t-1/2)-1}, \ldots\ldots k_1 k_0)$ and dealing with these two parts separately. An extra initial point is needed for processing the higher order part. The extra initial point is defined as $P_H = 2^{(t+1)/2} P$ to be consistent with the weight of the high – order part. Moreover, to reduce the number of 1's in the scalar operand k and to be conform with the operating sequence of the double- and-add algorithm, the operand k is encoded by complementary recoding. Note that two operating steps with the same step number but different suffixes, e.g $1a$ and $1b$, denote that these two operations can be executed independently. Algorithm 4.1 gives the algorithm.

Algorithm 5: Concurrent Algorithm with Complementary Recoding

Input: k and P

Output: $Q = kP$

$/*$ convert the integer k into the signed binary representation of the same using complementary recoding $*/$

$k' = \sum_{l=0}^{t-1} k_l 2^l, k_l \epsilon \{-1,0,1\}$
$Q = kP = k'_H P_H + \acute{k}_L P_L = Q_H + Q_L;$
$1a.\ Q_H = O, P_H = 2^{\lceil t/2 \rceil}$
$2a.\ for\ i\ from \frac{t}{2}\ to\ t-1\ do$
$3a.\ Q_H = Q_H + Q_H$
$4a.\ if\ k_i' = 1\ then\ Q_H = Q_H + P_H$

$1b. Q_L = O, P_L = P;$

$2b. for\ j\ from\ 0\ upto\ \left[\frac{t}{2}\right] - 1\ do$

$3b. Q_L = Q_L + Q_L;$

$4b. if\ k_j^{'} = 1\ then\ Q_L = Q_L + P_L$

$5. Q = Q_H + Q_L$

$return\ Q$

When applying the concurrent double –and-add algorithm using complementary recoding and drawing a time schedule similar to the one shown in 4.Algorithm, it reveals that one inversion circuit and two multipliers will be needed to accomplish the desired scalar multiplication, but the resulting hardware utilization is less than 100%. To overcome this deficiency, it is employed the division rather than the inversion circuit for the concurrent double-and-add algorithm. The table (a) gives comparison of timings of various scalar multiplication algorithms.

The proposed arithmetic operations have been implemented in the Xilinx Virtex300 FPGA using the Synopsys FPGA Express synthesis tool and Foundation 3.3i implementation tool. The functionality has been verified in the ModelSim simulator. Timings of various scalar multiplication algorithms (ms) is given below:

Table 1. Timings of Various Scalar Multiplication Algorithms (Ms)

Algorithms	**Binary**	**NAF**	**MOF**	**Proposed**
Timings (ms)	**1.769**	**1.051**	**0.884**	**0.667**

The timings of various scalar multiplication algorithm given in table 1. are based on the elliptic curve group of elements in $y^2 + y = x^3$ over $GF(2^5)$.

A point $(5, 21)$ means $(\alpha^5, \alpha^{21})$where α is the root of polynomial$x^5 + x^2 + 1$. There are 33 points (including the point at infinity) on the elliptic curve, out of which 10 point have order 11 while remaining 21 points have order 33. The cyclic group of the point $(5, 21)$ is shown in the following table. $2P = (20,7)$, $3P = (22,12)$, etc.,

$(\mathbf{5}, \mathbf{21})$	(20,7)	(22,12)	(18,26)	(27,1)	(26,30)	(23,5)
$(\mathbf{10}, \mathbf{19})$	(13,24)	(15,10)	$(-\infty, -\infty)$	(11,6)	(29,9)	(30,8)
$(\mathbf{21}, \mathbf{3})$	(9,13)	(9,14)	(21,29)	(30,20)	(29,16)	(11,27)
$(-\infty, \mathbf{0})$	(15,4)	(13,15)	(10,11)	(23,2)	(26,17)	(27,18)
$(\mathbf{18}, \mathbf{28})$	(22,23)	(20,22)	(5,25)			

5. CONCLUSION

Finite field $GF(2^5)$ arithmetic operations include addition, subtraction, multiplication, squaring and inversion. Due to proposed field $GF(2^5)$ and irreducible polynomial $f(x) = x^5 + x^2 + 1$ both additions and subtractions can be implemented very efficiently. Multiplication in PB using the said polynomial is 17% faster than RSA. Squaring a special case of multiplication can be implemented 40% faster than multiplication in the PB. Also inversions in the PB with the 'almost inverse method runs' 10% faster then RSA. Also this paper presents a concurrent algorithm using complementary recoding to speed up the scalar multiplication for the elliptic curve cryptosystem. With only an extra memory space to store an intermediate point, the algorithm can achieve 100% Hardware utilization based on the presented time schedule. Compared to the previous works, the time complexity of completing the scalar multiplication in this work can save 35%. In our future work, we are planning to reduce further the timings taken by scalar multiplication using other scalar recoding by reducing Hamming weight.

6. ACKNOWLEDGEMENT

The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

7. REFERENCES

[1] Blake, I., Seroussi, G., and Smart, N. Press, 1999. "Elliptic Curves in Cryptography, Cambridge University".

[2] Buhler, P., Lenstra, H.W., and Pomerance, C. "The development of the number field sieve", lecture Notes in Computer Science, volume- 1554, Springer-Verlag, 1994.

[3] W.Diffie and M.E.Hellman. 1976, "New directions in cryptography," IEEE Trans. On Information Theory, IT-22, (644-654).

[4] D.M.Gordon, 1998 "A Survey of fast exponention methods," J. Algorithms, 27, (129-146).

[5] Han, Y., leong, P., Tan, P., and Zhang, J "Fast Algorithms for Elliptic Curve Cryptosystems over Binary Finite Field," Advances in Cryptology-CRYPTO 1999. LNCS 1716, (75-85).

[6] Guajardo, J. and Paar, C. "Efficient Algorithms for Elliptic Curve Cryptosystems," Advances in Cryptology-CYYPTO 1997.LNCS 1462, (342-356).

[7] Itoh, T. and Tsujii, S. 1988, "A fast Algorithm for Computing Multiplication inverses in$GF(2^m)$ Using Normal Bases," Information & Computation, Volume-78, (171-177).

[8] Leca, C.L. and Rîncu C.I, May 2014, "Combining point operations for efficient elliptic curve cryptography scalar multiplication", 10th International Conference on Communications (1 – 4).

[9] Kobayashi, T. Morita, H., Kobayashi, K. and Hoshino, F. "Fast Elliptic Curve Algorithm Combining Frobenius map and Table referenced to Adapt to higher Characteristic," Adapt to higher Charecteristic," Advances in Cryptology-CRYPTO 1999. LNCS 1592, (176-189).

[10] ZhiLi, John Higgins, and Mark Element "Perfomance of Finite Field Arithmatic in an Elliptic Curve Crypto Systems" IEEE, (249-256), 0-7695-1315-8/01, 2001.

[11] Miller, V.S. "Use of Elliptic Curves in Cryptography," Advances in Cryptology CRYPTO'85, LNCS 218, Springer-Verlag, 1986, (417-426).

[12] Christina Thomas, "Analysis of Elliptic Curve Scalar Multiplication in Secure Communications". Global Conference on Communication Technologies (623-627), 2015.

[13] N.Koblitz, "Introduction to Elliptic Curves and Modular Forms", 2nd, Spinger-Verlag, 1993. N.Koblitz, "Elliptic Curve Cryptosystems", Math.Compu.Volume-48, No-177, January1987, (203-209).

[14] Chang, C.C, Kuo, Y.T. and Lin, C.H. March 2003. "Fast algorithms for common multiplicand multiplication and exponentiation, by performing complements", in Proceeding of 17th International Conference on Advanced Information Networking and Applications, (807-811).

[15] MuthuKumar, B. and Jeevananthan, S. 2010, "High Speed Hardware Implementation of an Elliptic Curve Cryptography (ECC) Co- Processor", Conference on Trendz in Information Sciences and Computing (176-180).

[16] D.E.Knuth, Seminumerical Algorithms, MA, Addison-Wesley 1981.

[17] Frey, G. "Applications of arithmetical geometry to cryptographic construction". Proceedings of the Fifth International Conference on Finite Fields and Applications, Springer-Verlag, 2001, (128-161).

[18] Galbraith, S., and Smart, N. 1999. "A cryptographic application of Weil descent", Codes and Cryptography, Lecture Notes in Computer science, 1746, springer-Velag, (191-200).

[19] Frey, G. "How to disguise an elliptic curve (Weil descent)" Talk at ECC'98, Waterloo, 1998.

[20] Gallant, R., Lambert, R. and Vanstone, S. 2000, "Improving the parallelized Pollard lambda search on anomalous binary curves", Mathematics of Computation, volume-69, (1699-1705).