Journal of Computer Science 2 (5): 395-400, 2006
ISSN 1549-3636
© 2006 Science Publications

Highly Efficient Elliptic Curve Crypto-Processor with Parallel GF(2^m) Field Multipliers

[1]Turki F. Al-Somani, [2]M. K. Ibrahim and [1]Adnan Gutub
[1]King Fahd University of Petroleum and Minerals, Computer Engineering Department
Dhahran 31261, Saudi Arabia
[2]De Montfort University, School of Engineering and Technology, Leicester LE19BH, UK

Abstract: This study presents a high performance GF(2^m) Elliptic Curve Crypto-processor architecture. The proposed architecture exploits parallelism at the projective coordinate level to perform parallel field multiplications. In the proposed architecture, normal basis representation is used. Comparisons between the Projective, Jacobian and Mixed coordinate systems using sequential and parallel designs are presented. Results show that parallel designs using normal basis gives better area-time complexity (AT^2) than sequential designs by 33-252% which leads to a wide range of design tradeoffs. The results also show that mixed coordinate system is the best in both sequential and parallel designs and gives the least number of multiplications levels when using 3 multipliers and the best AT^2 when using only 2 multipliers.

Key words: Elliptic curves cryptosystems, projective coordinate, parallel designs, normal basis

INTRODUCTION

Recently, Elliptic Curves Cryptosystems (ECC)[1,2] has attracted many researchers and has been included in many standards[3-8]. ECC is evolving as an attractive alternative to other public-key schemes such as RSA by offering the smallest key size and the highest strength per bit. Extensive research has been done on the underlying math, security strength and efficient implementations. Among the different fields that can underlie elliptic curves, prime fields GF(p) and binary polynomial fields GF(2^m) have shown to be best suited for cryptographic applications. In particular, binary fields allow for fast computation in software as well as in hardware. Small key sizes and computational efficiency make ECC not only applicable to hosts processing security protocols over wired networks, but also to small wireless devices such as cell phones, PDAs and Smartcards.

Inversion operations, which are needed in point addition over Elliptic Curves are the most expensive operation over Finite Fields[9-12]. The approach adopted in the literature is to represent Elliptic Curve points in projective coordinate in order to replace the inversion operations with repetitive multiplications[9-15]. Recently, several ECC processors have been proposed in the literature[10-12,14,15] based on projective coordinate representation. There are many projective coordinate systems to choose from. In exiting architectures, the selection of a projective coordinate is based on the number of arithmetic operations, mainly multiplications. This is to be expected due to the sequential nature of these architectures where a single multiplier is used.

For high performance servers, such sequential architectures are too slow to meet the demand of increasing number of users. For such servers, high-speed crypto processors are becoming crucial. One solution for meeting this requirement is to exploit the inherent parallelism within Elliptic curve point operations in projective coordinate. Recently, ECC processor architectures have been proposed where the choice of the projective coordinate system used also depends on its inherent parallelism[11,12]. Since multiplication is the most dominant operation and most time consuming when computing point operations in projective coordinate, three multipliers that can work in parallel are used in the architectures in[11,12]. These architectures give better area-time complexity (AT^2) than the architectures that are based in a single multiplier. In this study we are proposing an alternative parallel design using normal basis representation which is more suitable for hardware implementations. In addition, the complexity and parallelism in several homogenous and heterogeneous projective coordinate are given.

GF(2^m) Arithmetic background: The finite GF(2^m) field has particular importance in cryptography since it leads to particularly efficient hardware implementations. Elements of the field are represented in terms of a basis. Most implementations use either a Polynomial Basis or a Normal Basis[16]. For the proposed cryptoprocessor described in this study, a normal basis is chosen since it leads to more efficient hardware implementations. Normal basis is more suitable for hardware implementations than polynomial basis since operations are mainly comprised of rotation,

Corresponding Author: Turki F. Al-Somani, King Fahd University of Petroleum and Minerals, Computer Engineering Department, Dhahran 31261, Saudi Arabia

shifting and exclusive-OR operations which can be efficiently implemented in hardware. A normal basis of GF(2^m) is a basis of the form
($ß, ß^2, ß^4, ß^8, \ldots ß^{2^{(m-1)}}$) , where $ß \in$ GF(2^m)
In a normal basis, an element $A \in$ GF(2^m) can be uniquely represented in the form

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i},$$

where $a_i \in \{0, 1\}$.
GF(2^m) operations using normal basis are performed as follows:

Addition and subtraction: Addition and subtraction are performed by a simple bit-wise exclusive-OR (XOR) operation.

Squaring: Squaring is simply performed by a rotate left operation.

Multiplication: $\forall$A, B $\in$ GF(2^m), where

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$$

and

$$B = \sum_{i=0}^{m-1} b_i \beta^{2^i},$$

the product C = A*B, is given by:

$$C = A * B = \sum_{i=0}^{m-1} c_i \beta^{2^i}$$

then multiplication is defined in terms of a multiplication table $\lambda_{ij} \in \{0, 1\}$

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_{ij} a_{i+k} b_{j+k} \tag{1}$$

An optimal normal basis (ONB)[17] is one with the minimum number of terms in (2.1), or equivalently, the minimum possible number of nonzero λ_{ij}. This value is 2m-1 and since it allows multiplication with minimum complexity, such a basis would normally lead to more efficient hardware implementations.

Inversion: Inverse of a $\in$ GF(2^m), denoted as a^{-1}, is defined as follows.

$aa^{-1} = 1 \mod 2^m$

Most inversion algorithms used are derived from Fermat's Little Theorem:

$$a^{-1} = a^{2^m - 2} = (a^{2^{m-1}-1})^2$$

for all a $\neq$ 0 in GF(2^m). Itoh and Tsujii inversion algorithm[18], however, is one of the most efficient inversion algorithms that have been proposed thus far.

Elliptic curves: Here we present a brief introduction to elliptic curves. Let GF(2^m) be a finite field of characteristic two. A non-supersingular elliptic curve E over GF(2^m) is defined to be the set of solutions (x, y) $\in$ GF(2^m) X GF(2^m) to the equation, $y^2 + xy = x^3 + ax^2 + b$, where a and b $\in$ GF(2^m), b $\neq$ 0, together with the point at infinity denoted by O. It is well known that E forms a commutative finite group, with O as the group identity, under the addition operation known as the tangent and chord method. Explicit rational formulas for the addition rule involve several arithmetic operations (adding, squaring, multiplication and inversion) in the underlying finite field. In affine coordinate, the elliptic group operation is given by the following.

Let P = (x_1, y_1) $\in$ E; then -P = (x_1, $x_1 + y_1$). For all P $\in$ E, O + P = P + O = P. If Q = (x_2, y_2) $\in$ E and Q $\neq$ -P, then P + Q = (x_3 , y_3),
where

$$x_3 = (\frac{y_1 + y_2}{x_1 + x_2})^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a$$

$$y_3 = (\frac{y_1 + y_2}{x_1 + x_2}) \cdot (x_1 + x_3) + x_3 + y_1$$

if P $\neq$ Q and

$$x_3 = x^2{}_1 + \frac{b}{x^2{}_1}$$

$$y_3 = x^2{}_1 + (x_1 + \frac{y_1}{x_1}) x_3 + x_3$$

if P = Q.

Computing P + Q is called elliptic curve point addition if P $\neq$ Q and is called elliptic curve point doubling if P = Q.

Scalar multiplication is the basic operation for ECC. Scalar multiplication in the group of points of an elliptic curve is the analogous of exponentiation in the multiplicative group of integers modulo a fixed integer m. Computing dP can be done with the straightforward double-and-add approach based on the binary expression of d = (d_{l-1},…,d_0) where d_{l-1} is the most significant bit of d. However, several scalar multiplication methods have been proposed in the literature. A good survey is presented by Gordon in[19].

Projective coordinate in GF(2^m): The projective coordinate are used to eliminate the need for performing inversion. For elliptic curve defined over GF(2^m), many different forms of formulas are found[9,20,22] for point addition and doubling. The projective coordinate system (Pr), so called homogeneous coordinate system, have the form (x,y)=(X/Z,Y/Z)[20], while the Jacobian coordinate system have the form (x,y)=(X/Z^2,Y/Z^3)[9]. From the Jacobian coordinate system, two other coordinate systems where proposed. These are: the Chudnovsky Jacobian coordinate system (J^c) representing the point with the quintuple (X, Y, Z, Z^2, Z^3) and the Modified Jacobian coordinate system (J^m) representing the point with the quadruple (X, Y, Z, aZ^4). Mixed coordinate was proposed in[22] leading to better performance. Table 1 demonstrates only the multiplications needed in the

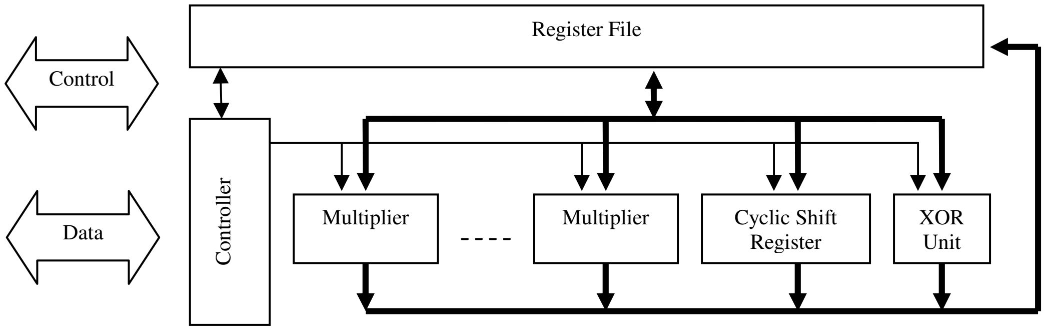

Fig. 1: The proposed architecture

Table 1: Multiplications within different coordinate systems

Projective Coordinate (Pr)				Jacobian Coordinate (J)			
Addition		Doubling		Addition		Doubling	
$A = X_1Z_2$	1M	$A=X_1Z_1$	1M	$A = X_1Z_2^2$	1M	$Z_3=X_1Z_1^2$	1M
$B = X_2Z_1$	1M	$B= bZ_1^4+X_1^4$	1M	$B = X_2Z_1^2$	1M	$A = bZ_1^2$	1M
$C = A+B$		$C= AX_1^4$	1M	$C = A+B$		$B = X_1+A$	
$D = Y_1Z_2$	1M	$D=Y_1Z_1$	1M	$D = Y_1Z_2^3$	2M	$X_3 = B^4$	
$E = Y_2Z_1$	1M	$E=X_1^2+D+A$		$E = Y_2Z_1^3$	2M	$C = Z_1Y_1$	1M
$F = D+E$		$Z_3=A^3$	1M	$F = D+E$		$D=Z_3+X_1^2+C$	
$G= C+F$		$X_3=AB$	1M	$G = Z_1C$	1M	$E = DX_3$	1M
$H= Z_1Z_2$	1M	$Y_3= C+BE$	1M	$H = FX_2+GY_2$	2M	$Y_3 = X_1^4Z_3+E$	1M
$I=C^3+aHC^2+HFG$	5M			$Z_3 = GZ_2$	1M		
$X_3 = CI$	1M			$I =F+Z_3$			
$Z_3 = HC^3$	1M			$X_3= aZ_3^2+IF+C^3$	3M		
$Y_3=GI+C^2[FX_1+CY_1]$	4M			$Y_3= IX_3+HG^2$	2M		
Total	16M		7M		15M		5M

Table 2: Multiplication cycles for the coordinate systems

	Critical Path		Average No. of Multiplication cycles			
Coordinate System	Addition	Doubling	1 Multiplier	2 Multipliers	3 Multipliers	4 Multipliers
Projective Coordinate	4	2	15	8	6	4
Jacobian Coordinate	5	2	12.5	7	5	4.5
Mixed Coordinate	3	2	7	4	3.5	3.5

Table 3: Multiplication cycles within mixed coordinate system

Addition	No. of Multipliers				Doubling	No. of Multipliers			
	1	2	3	4		1	2	3	4
$t(J^m+ J^m)$	16M	8M	6M	5M	t(2Pr)	7M	4M	3M	2M
$t(J^m+ J^c= J^m)$	15M	8M	5M	5M	$t(2J^c)$	5M	3M	2M	2M
$t(J+ J^c= J^m)$	15M	8M	6M	5M	t(2J)	5M	3M	2M	2M
t(J+ J)	15M	8M	6M	5M	$t(2J^m= J^c)$	6M	3M	2M	2M
t(Pr+ Pr)	16M	8M	6M	5M	$t(2J^m)$	6M	3M	2M	2M
$t(J^c+ J^c= J^m)$	15M	8M	5M	5M	$t(2A= J^c)$	4M	2M	2M	2M
$t(J^c+ J^c)$	15M	8M	6M	5M	$t(2J^m= J)$	5M	3M	2M	2M
$t(J^c+ J= J)$	14M	7M	6M	5M	$t(2A= J^m)$	4M	2M	2M	2M
$t(J^c+ J^c= J)$	14M	7M	6M	5M	t(2A= J)	3M	2M	2M	2M
$t(J+ A= J^m)$	12M	6M	4M	4M					
$t(J^m+ A= J^m)$	12M	6M	4M	4M					
$t(J^c+ A= J^m)$	12M	6M	4M	4M					
$t(J^c+ A= J^c)$	12M	6M	4M	4M					
t(J+ A= J)	11M	6M	4M	4M					
$t(J^m+ A= J)$	11M	6M	4M	4M					
$t(A+ A= J^m)$	8M	4M	3M	3M					
$t(A+ A= J^c)$	8M	4M	3M	3M					

Projective and Jacobian coordinate systems since other field arithmetic operations requires negligible time as compared to multiplication. This is because of the nature of normal basis over GF(2^m) which performs addition and subtraction simply by an XOR operation and performs squaring by a single rotation as pointed earlier.

ECC Crypto-processor architecture: This section defines the basic idea and the proposed generic architecture of the ECC crypto-processor. Also, the methodology used to find the number of multipliers in each parallel design will be discussed.

Generic ECC Crypto-processor architecture with multi-multipliers: The basic idea is based on the parallelism of projective coordinate multiplications proposed in[11,12]. Three multipliers were employed to provide parallelism to provide better AT^2.

The work reported in[11,12] was represented in polynomial basis and squaring was considered to be a multiplication, which can be negligible in normal basis or when using irreducible trinomial[21]. This makes a big difference in the number of multiplication cycles as is discussed in the next section. The proposed generic crypto-processor architecture is based on normal basis and uses 2-4 multipliers, a cyclic shift register to perform squaring, an XOR unit for field addition and a register file. Only one cyclic shift register and XOR unit is used since both squaring and filed addition requires only one clock cycle and hence it can be reused several times while a single multiplication operation is computed. Each of these arithmetic units can get operands from the register file and store the result in the register file. The controller generates control signals for all the arithmetic units and the register file (Fig. 1).

Methodology used to find the number of multipliers: Since multiplication is the dominant operation in elliptic curve point operations in projective coordinate and since the computation time of multiplication is much higher than field squaring and addition, the emphasis in this study is to speed up the computations of point operations in projective by performing more than one multiplication operation at any one time.
The approach adopted in this study is:

1. Analyzing the dataflow of point operations for each projective coordinate system in the following manner:
   - ❖ Find the critical path which has the lowest number of the multiplication operations.
   - ❖ Find the maximum number of multipliers that are needed to meet this critical path.
2. Varying the number of multipliers from one to the number of multipliers specified by the critical path to find the following:
   - ❖ Find the best schedule of each dataflow using the specified number of multipliers.
   - ❖ Find the AT^2 .

The critical paths of the Projective and Jacobian coordinate systems are listed in Table 2 for both the point addition and doubling. Mixed coordinate system's critical path is chosen as the best critical path than can be reached among all other mixed coordinate systems. The critical path of the Projective coordinate system is 4 and 3 for point addition and doubling respectively. From Table 1, we can see that the total number of multiplications needed with the projective coordinate system is 16 and 7 for point addition and doubling respectively. This means that using one multiplier gives an average of (16/2) + 7 = 15 multiplications cycles since, on average, we perform doubling for all the bits in the key and perform point addition only for half of the key bits.

Table 2 summarizes the average number of multiplications cycles required for point operations using 1, 2, 3 and 4 multipliers and Table 3 shows clearly the advantage of using parallel designs reducing the average number of multiplications cycles when using Mixed coordinate system. It is worth noting that unlike the work reported in[11,12] where polynomial basis is used and squaring was considered to be a multiplication, which can be negligible when using normal basis or when using irreducible trinomial[21]. This makes a big difference in the number of multiplication cycles as can be seen from Table 2 and 3 and also has a significant impact on the utilization of multipliers.

RESULTS

In Table 4, comparisons between the different coordinate system are shown. Four cases are covered in these comparisons:
Single multiplier (Sequential), Two, multipliers (Parallel), Three multipliers (Parallel) as in[11,12] and Four multipliers (Parallel).

The results in Table 4 show that the parallel designs are always giving better AT^2 than the sequential design by 33-252% (Table 5). This wide range of enhancements provides the designers with large range of trade-offs.

It is clear from Table 4 that with the Projective coordinate system, the enhancement in the AT^2 increases by employing more multipliers. The maximum number of multipliers that can be reached that satisfies the critical path was found to be 4 multipliers. The enhancements using parallel designs with the Projective coordinate system, as shown in Table 5, was found to be 76%, 108% and 252% when using 2, 3 and 4 multipliers respectively. However, the Projective coordinate system was giving better AT^2 than Jacobian coordinate system when employing 4 multipliers, while it was giving worse results by using less number of multipliers.

Only the Jacobian projective coordinate system can benefit from using 5 multipliers and requires an average of 4 multiplication cycles which is the same as what the

Table 4: Comparison between the different designs

Coordinate System	No. of multipliers (A)	No. of Cycles for multiplications	Time, nsec (T)	AT^2
Projective	1	15	15.00	225.00
	2	8	8.00	128.00
	3	6	6.00	108.00
	4	4	4.00	64.00
Jacobian	1	12.5	12.50	156.25
	2	7	7.00	98.00
	3	5	5.00	75.00
	4	4.5	4.50	81.00
Mixed	1	7	7.00	49.00
	2	4	4.00	32.00
	3	3.5	3.50	36.75
	4	3.5	3.50	49.00

Table 5: Comparison between the different designs based on Table 4

			Enhancement Percentage %			
			Number of Multipliers			
Coordinate System	Number of Multipliers	AT^2	1	2	3	4
Projective	1	4257.56	-	-	-	-
	2	2422.08	0.76	-	-	-
	3	2043.63	1.08	0.19	-	-
	4	1211.04	2.52	1.00	0.69	-
Jacobian	1	2956.64	-	-	-	-
	2	1854.41	0.59	-	-	-
	3	1419.19	1.08	0.31	-	0.08
	4	1532.72	0.93	0.21	-	-
Mixed	1	927.2	-	-	-	-
	2	605.52	0.53	-	0.15	0.53
	3	695.4	0.33	-	-	0.33
	4	927.2	-	-	-	-

Projective coordinate gives with only 4 multipliers. Also, we can notice that using 3 multipliers, as in[11,12], is giving better result than using 4 multipliers with the Jacobian coordinate system (Table 4). This shows clearly that adding more multipliers does not necessarily increase performance as depicted in Fig. 2.

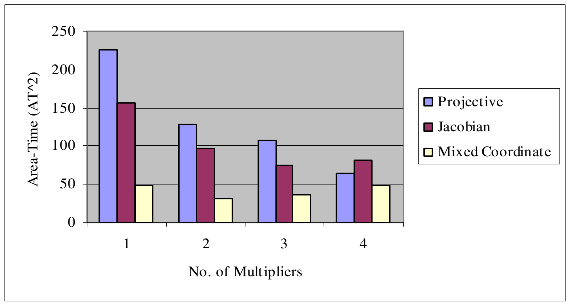

Fig. 2: Comparison between the different designs

However, the best results reported in Table 4 were found to be when using the Mixed coordinate system. It is clearly obvious that Mixed coordinate is giving always the best AT^2 as compared to others. It also can be easily seen from Table 2 that using 4 multipliers will give the same multiplications cycles as when using only 3 multipliers. From Table 4 and 5, we can see that 2 multipliers give absolutely the best AT^2 in comparison to all other implementations including the use of a single multiplier. What is a more significant observation from Table 4 and 5 is that using the proposed architecture with Mixed coordinate system is not only faster for parallel implementation but it also leads to a better AT^2 (cost) than other alternatives.

CONCLUSION

In this study we presented a high performance $GF(2^m)$ Elliptic Curve Crypto processor. Parallelism was exploited at the projective coordinate level using 2, 3 and 4 multipliers to perform parallel field multiplications represented in optimal normal basis. Comparisons between the Projective, Jacobian and Mixed coordinate systems using sequential and parallel designs was also presented. The results show that using parallel designs in optimal normal basis gives better AT^2 than sequential designs by almost 33-252% which gives the designers a wide large of design tradeoffs. The results also show that mixed coordinate are the best in both sequential and parallel designs and gives the least multiplications cycles using 3 multipliers and the best AT^2 with only 2 multipliers.

ACKNOWLEDGMENT

The authors would like to acknowledge the support King Fahd University of Petroleum and Minerals (KFUPM).

REFERENCES

1. Koblitz, N., 1987. Elliptic curve cryptosystems. Math. of Comput., 48: 203-209.
2. Menezes, A.J., 1993. Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers.
3. ANSI X9.62, 1998. Public Key Cryptography for the Financial Services Industry: Curve Digital Signature Algorithm (ECDSA).
4. IEEE P1363, IEEE Standard Specifications for Public-Key Cryptography, 2000.
5. National Institute of Standards and Technology, 2000. Recommended elliptic curves for federal government use. Appendix to FIPS 186-2.
6. Standards for Efficient Cryptography Group/Certicom Research, 200. SEC 1: Elliptic curve cryptography. Version 1.0. http://www.secg.org/.
7. Standards for Efficient Cryptography Group/Certicom Research, 2000. SEC 2: Recommended elliptic curve cryptography domain parameters. Version 1.0. http://www.secg.org/.
8. Wireless Application Protocol (WAP) Forum, Wireless Transport Layer Security (WTLS) Specification. http://www.wapforum.org/.
9. Blake, I., G. Seroussi and N. Smart, 1999. Elliptic Curves in Cryptography. Cambridge University Press: New York.
10. Gutub, A.A., A.F. Tenca, E. Savas and C.K. Koc, 2002. Scalable and unified hardware to compute Montgomery inverse in GF(p) and GF(2^n). Cryptographic Hardware and Embedded Systems.
11. Gutub, A.A. and M.K. Ibrahim, 2003. High radix parallel architecture for GF(p) elliptic curve processor. Proc. IEEE Conf. on Acoustics, Speech and Signal Processing ICASSP 2003, pp: 625-628, Hong Kong.
12. Gutub, A.A. and M.K. Ibrahim, 2003. High performance elliptic curve GF(2^k) cryptoprocessor architecture for multimedia. Proc. IEEE Intl. Conf. on Multimedia and Expo, ICME 2003, pp: 81-84, Baltimore, Maryland, USA.
13. Miyaji, A., 1992. Elliptic curves over F_P suitable for cryptosystems, advances in cryptology, Australia.
14. Okada, S., N. Torii, K. Itoh and M. Takenaka, 2000. Implementation of elliptic curve cryptographic coprocessor over GF(2^m) on an FPGA. Workshop on Cryptographic Hardware and Embedded Systems, Massachusetts.
15. Stinson, D.R., 1995. Cryptography: Theory and Practice. CRC Press, Boca Raton, Florida.
16. Lidl, R. and H. Niederreiter, 1994. Introduction to finite fields and their applications. Cambridge University Press, Cambridge, UK.
17. Mullin, R.C., I.M. Onyszchuk, S.A. Vanstone and R.M. Wilson, 1989. Optimal normal bases in GF(p^m). Discrete Appl. Math., 22: 149-161.
18. Itoh, T. and S. Tsujii, 1988. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. Info. Comput., 78: 171-177.
19. Gordon, D., 1998. A survey of fast exponentiation methods. J. Algorithms, pp: 129-146.
20. Ernst, M., Klupsch, Hauck and Huss, 2001. Rapid prototyping for hardware accelerated elliptic curve public-Key cryptosystems. The IEEE 12th Intl. Workshop on Rapid System Prototyping, Monterey, CL.
21. Wu, H., 2002. Bit-parallel finite field multiplier and squarer using polynomial basis. IEEE Trans. Comp., 51: 750-758.
22. Cohen, H., T. Ono and A. Miyaji, 1998. Efficient elliptic curve exponentiation using mixed coordinate. In Advances in Cryptology-SIACRYPT. K. Ohta and D. Pei, Eds., Lecture Notes in Computer Science, 1514: 51-65.