High Performance Elliptic Curve GF(2^m) Cryptoprocessor Secure Against Timing Attacks

Turki F. Al-Somani[†] *and M. K. Ibrahim*[††]

[†]King Fahd University of Petroleum & Minerals, Computer engineering Department,
P.O. Box: 602, Dhahran 31261, Saudi Arabia
[††]De Montfort University, School of Engineering & Technology, Leicester LE19BH, UK

Summary
This paper presents a high performance GF(2^m) Elliptic Curve Cryptoprocessor architecture. The proposed cryptoprocessor is based on normal basis representation and uses three multipliers to perform parallel field multiplications. Point operations are performed using Mixed coordinate system to increase the performance and the immunity against timing attacks. The basic idea is to select a combination of point addition and point doubling from Mixed coordinate system such that both point operations requires the same number of multiplication cycles. Thus, an attacker cannot distinguish between point doubling and point addition and therefore it is not possible to extract the key pattern using a timing attack. Results show that the proposed cryptoprocessor gives better time complexity than existing designs which use fake computations by 76%. The proposed cryptoprocessor has been synthesized on a Xilinx Vertex II FPGA (xc2v8000) over GF(2^{173}) and it required 159522 clock cycles to perform scalar multiplication. The proposed cryptoprocessor required 28154 Slices, which is only 60% out of the total number of available Slices.

Key words:
Elliptic Curves Cryptosystems, Mixed Coordinates, Parallel Designs, Normal Basis.

1. 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.

Side channel attacks on such devices are considered serious threats due to the physical characteristics of these devices and their use in potentially hostile environments. Side channel attacks seek to break the security of these devices through observing their power consumption trace or computations timing [9]. Careless or naive implementations of cryptosystems may allow side channel attacks to infer the secret key or obtain partial information about it. Thus, designers of such systems seek to introduce algorithms and designs that are not only efficient, but also side channel attack resistant.

Inversion operations, which are needed in point addition over Elliptic Curves are the most expensive operation over Finite Fields [10]. 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. Recently, several ECC processors have been proposed in the literature 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 cryptoprocessors are becoming crucial. One solution for meeting this requirement is to exploit the inherent parallelism within Elliptic curve point operations in projective coordinate.

This paper presents a high performance elliptic curve cryptoprocessor over GF(2^m). Parallelism is exploited at the projective coordinate level to increase both the performance and the immunity against timing attacks. The rest of this paper is organized as follows:

Manuscript received January, 2006.

Section 2 explains the GF(2^m) arithmetic background. Section 3 gives a brief introduction to ECC. Section 4 discusses the projective coordinates in GF(2^m). The proposed architecture will be explained in Section 5. Section 6 discusses and compares the efficiency for the existing and the proposed architecture. Finally, Section 7 concludes this work.

2. 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 [11]. For the implementation described in this paper, 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, 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

$(\beta^{2^{(m-1)}}, \ldots\ldots, \beta^8, \beta^4, \beta^2, \beta)$, where $\beta \in$ GF(2^m)

In a normal basis, an element $A \in$ GF(2^m) can be uniquely represented in the form

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

where $a_i \in \{0, 1\}$.

GF(2^m) operations using normal basis are performed as follows:

1. **Addition and Subtraction**: Addition and subtraction are performed by a simple bit-wise exclusive-OR (XOR) operation.
2. **Squaring:** Squaring is simply performed by a rotate left operation.
3. **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:*

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

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

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_{ij} a_{i+k} b_{j+k} \quad (3)$$

An optimal normal basis (ONB) [12] is one with the minimum number of terms in (3), 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.

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

$$a a^{-1} = 1 \mod 2^m \quad (4)$$

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

$$a^{-1} = a^{2^m - 2} = (a^{2^{m-1}-1})^2 \quad (5)$$

for all $a \neq 0$ in GF(2^m).

3. 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,

$$y^2 + xy = x^3 + ax^2 + b, \quad (6)$$

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 coordinates, 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

$$x_3 = (\frac{y_1 + y_2}{x_1 + x_2})^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a \quad (7)$$

$$y_3 = (\frac{y_1 + y_2}{x_1 + x_2}) \cdot (x_1 + x_3) + x_3 + y_1 \quad (8)$$

if $P \neq Q$ and,

$$x_3 = x^2{}_1 + \frac{b}{x^2{}_1} \quad (9)$$

$$y_3 = x^2{}_1 + (x_1 + \frac{y_1}{x_1})x_3 + x_3 \quad (10)$$

if $P = Q$.

Computing $P + Q$ is called elliptic curve point addition if $P \neq Q$ and is called elliptic curve point doubling if $P = Q$. Point subtraction is a useful operation in some algorithms. This operation can be performed with the point addition or point doubling formulas using the additive inverse of the point to be subtracted. For example, the point subtraction $P - Q$ can be computed using the point addition operation where: $P - Q = P + (-Q)$. The additive inverse of a point $P = (x, y)$ is the point $(x, x + y)$ for curves defined over the GF(2^m) fields.

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, as described in Algorithm 1, based on the binary expression of $d = (d_{l-1},\ldots,d_0)$ 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 [13].

Algorithm 1: from most significant bit

input P, d
$Q \leftarrow P$
for i from $l-2$ to 0 do
    $Q \leftarrow 2Q$
    if $d_i = 1$ then $Q \leftarrow Q + P$
output Q

In the NAF method [14], signed digit representation are used ($d = \sum_{i=0}^{l-1} d_i 2^i$), where $d_i \in \{0, \pm 1\}$. NAF has the property that no two consecutive coefficients d_i are nonzero. Every positive integer d has a unique NAF, denoted NAF(d). Moreover, NAF(d) has the fewest nonzero coefficients of any signed digit representation of d (see Algorithm 2).

Algorithm 2 (Binary NAF method): from most significant bit

input P, NAF(d)= $\sum_{i=0}^{l-1} d_i 2^i$
$Q \leftarrow O$
for i from $l-1$ to 0 do
    $Q \leftarrow 2Q$
    if $d_i = 1$ then $Q \leftarrow Q + P$
    if $d_i = -1$ then $Q \leftarrow Q - P$
output Q

4. Projective Coordinate in GF(2^m)

The projective coordinate are to eliminate the need for performing inversion. For elliptic curve defined over GF(2^m), many different forms of formulas are found [9] for point addition and doubling. The projective coordinate system (Pr), so called homogeneous coordinates, have the form $(x,y)=(X/Z,Y/Z)$, while the Jacobian coordinate system have the form $(x,y)=(X/Z^2,Y/Z^3)$. From the Jacobian coordinates, two other coordinates 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 system was proposed in [15] leading to better performance. In Mixed coordinate system, point operations can be performed using different coordinate systems. Table 1 demonstrates all possible efficient combinations of different coordinate systems using Mixed coordinate system over GF(2^m). Point addition using Mixed coordinate system can have the first point represented in a coordinate system and the other point in another coordinate system and the result also in another coordinate system. For point addition in Table 1, if the two points are represented using the same coordinate system, the result will be in the same coordinate system. If the two points are represented with two different coordinate systems, the result may be the coordinate of one of the two point's coordinate system or another coordinate system.

Point doubling using Mixed coordinate system also can have the point in a coordinate system and the resulting point in the same coordinate system or in another coordinate system (see Table 1).

5. The Proposed Cryptoprocessor

The basic idea of the proposed architecture is exploit the parallelism at the projective coordinate level to increase both the performance and the resistance to timing attacks. Since multiplication is the most dominant operation and most time consuming when computing point operations in normal basis, point addition and point doubling are selected from Table 1 such that both point operations requires the same number of multiplication cycles. The dataflow of all combinations of Table 1 are investigated using parallel multipliers such that point addition and point doubling requires exactly the same multiplication cycles. This makes it impossible for an attacker to distinguish between point addition and point doubling via timing attacks.

Table 1: Mixed Coordinate System.

Addition	*Doubling*
$(J^m + J^m)$	$(2Pr)$
$(J^m + J^c = J^m)$	$(2J^c)$
$(J + J^c = J^m)$	$(2J)$
$(J + J)$	$(2J^m = J^c)$
$(Pr + Pr)$	$(2J^m)$
$(J^c + J^c = J^m)$	$(2A = J^c)$
$(J^c + J^c)$	$(2J^m = J)$
$(J^c + J = J)$	$(2A = J^m)$
$(J^c + J^c = J)$	$(2A = J)$
$(J + A = J^m)$	
$(J^m + A = J^m)$	
$(J^c + A = J^m)$	
$(J^c + A = J^c)$	
$(J + A = J)$	
$(J^m + A = J)$	
$(A + A = J^m)$	
$(A + A = J^c)$	

In the proposed ECC cryptoprocessor, point addition is performed using Mixed coordinate system using the combination $(J^c + A = J^m)$ which requires 4 multiplication cycles as illustrated in Fig. 1 where a circle represents a multiplication and rectangle represent either field addition or squaring. The input of point addition is a point represented in the Chudnovsky Jacobian (J^c) coordinate system and another point represented in affine coordinate system. The resulting point of point addition is represented in the Modified Jacobian (J^m) coordinate system.

On the other hand, the input of point doubling is selected to be in Modified Jacobian (J^m) coordinate system and the resulting point is represented in Chudnovsky Jacobian (J^c) coordinate system. This requires three multiplication cycles using three multipliers. However, we changed the dataflow scheduling to be performed within four multiplication cycles such that point addition and point doubling will require the same time exactly (see Fig. 2).

The proposed cryptoprocessor architecture uses 3 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 field 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 (see Fig. 3).

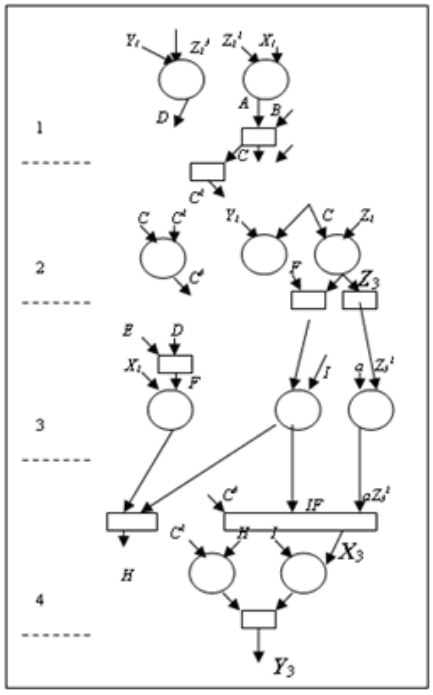

Fig. 1: Dataflow of point addition.

Several normal basis multipliers have been proposed in the literature. The most well known and widely used is the

Massey-Omura bit-serial multiplier [16]. The Massey-Omura bit-serial multiplier works for both the optimal normal basis of type I and type II. Massey-Omura multiplier is selected because it support both type I and type II optimal normal basis which makes the proposed architecture more parametric. Fig. 4 shows an example of the Massey-Omura bit-serial multiplier over GF(2^5). In the example shown in Fig. 4 we have

$$c_0 = a_4b_4 + (a_0b_1 + a_1b_0) + (a_1b_3 + a_3b_1) + (a_2b_4 + a_4b_2) + (a_2b_3 + a_3b_2) \quad (11)$$

In order to obtain c_1, simply rotate both A and B and use the same circuit.

$$c_1 = a_0b_0 + (a_1b_2 + a_2b_1) + (a_2b_4 + a_4b_2) + (a_3b_0 + a_0b_3) + (a_3b_4 + a_4b_3) \quad (12)$$

The space complexity of Massey-Omura multiplier is $(2m - 1)$ AND gates + $(2m - 2)$ XOR gates, while the time complexity is $T_A+(1+ \log_2(m-1))T_X$, where T_A and T_X are the time delay of an AND and an XOR gates respectively.

The proposed cryptoprocessor uses NAF method [14] for scalar multiplication. This increases the performance since the number of 1s in the private key will be on average 1/3 of the key bits instead of 1/2 of the key bits using the normal binary encoding. Accordingly, the number of point additions will be reduced.

6. Results and Comparisons

Chevallier-Mames *et. al.* in [17] proposed inserting fake computation within point doubling to make point doubling requires the same time of point addition which is very costly. The time complexity of the proposed work in [17] is *{[15*n + 15*(n/2)] * n}*, where point doubling and addition requires 15 multiplications using the Jacobian coordinate system in addition to the fake computations.

More recently, Hodjat *et. al.* in [18] proposed a new scalar multiplication algorithm that inspects three bits at a time. The proposed work in [18] defined new dataflow for both point addition and point doubling. The time complexity to perform scalar multiplication of the proposed design in [18] is *{[(18*(n+3) + (n+3)/2 +1)] * (n/3)}*. However, the proposed work in [18] uses two multipliers and a squarer using polynomial basis where squaring requires 1/2 the time required for a multiplication.

In this paper, the proposed cryptoprocessor is based on normal basis which squaring is implemented simply by a rotate operation which requires only one clock cycle. Instead of using two multipliers as in [18], three multipliers are used to perform parallel field multiplications. The proposed cryptoprocessor requires *{[4*n + 4*(n/3)] * n}* to perform scalar multiplication using NAF method. The proposed cryptoprocessor performs point doubling and addition within four multiplication cycles. The number of point doubling and addition using NAF method is *(n)* and *(n/3)* respectively.

Table 2 summarize the time complexities of the proposed designs in [17, 18] and the proposed design here. Fig. 5 shows the time complexity of these different designs. It is clear that the proposed cryptoprocessor here is the best among the others. The timing performance of proposed cryptoprocessor is better than the proposed design in [17] by 76% without losing the immunity against timing attacks. The proposed cryptoprocessor also performs better than the proposed design in [18] by 15%.

The proposed cryptoprocessor has been synthesized on a Xilinx Vertex II FPGA (xc2v8000) over GF(2^{173}). The total number of clock cycles on average, with 173 double and 58 add, required 159522 clock cycles and it required only 28154 out of 46592 Slices. The synthesis report showed that the minimum clock period is 18.753 ns (7.962 ns logic, 10.792 ns route), which means that the maximum clock frequency is 53.323 MHz.

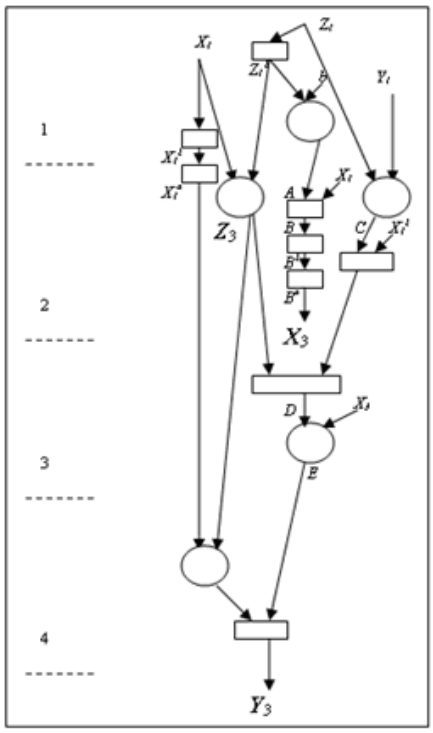

Fig. 2: Dataflow of point doubling.

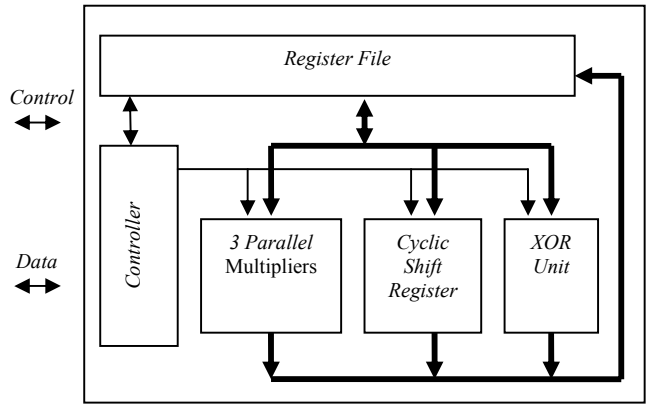

Fig. 3: The Proposed Architecture

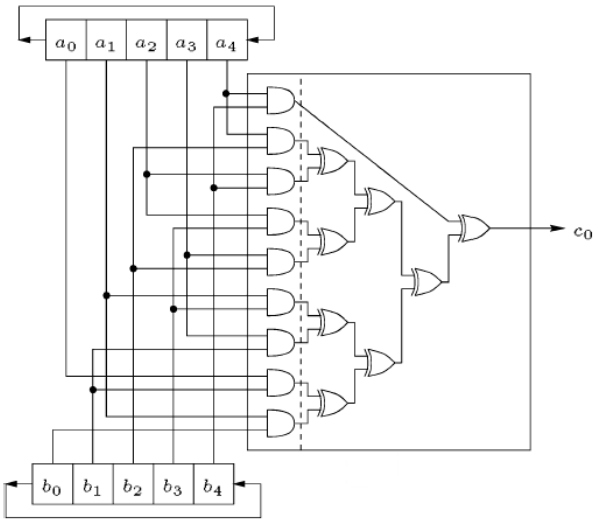

Fig. 4: Massey-Omura bit-serial multiplier over GF(2^5).

Table 2: The time complexity of different designs.

Reference Design	*Time Complexity*
Chevallier-Mames *et. al.* [17]	*{[15*n + 15*(n/2)] * n}*
Hodjat *et. al.* [18]	*{[(18*(n+3) + (n+3)/2 +1)] * (n/3)}*
Presented here	*{[4*n + 4*(n/3)] * n}*

7. Conclusion

In this paper we presented a high performance GF(2^m) Elliptic Curve Cryptoprocessor. The proposed cryptoprocessor is based on normal basis representation and uses three multipliers to perform parallel field multiplications. Point operations are performed using Mixed coordinate system to increase the performance and the immunity against timing attacks. The basic idea is based on performing point addition point doubling such that both point operations requires the same number of multiplication cycles. This is done by scheduling the dataflow of point addition and point doubling to be computed within four multiplication cycles. Thus, an attacker cannot distinguish between point doubling and point addition and therefore it is not possible to extract the key pattern using a timing attack.

Results show that the proposed cryptoprocessor gives better time complexity than existing designs that use extra dummy computations by 76%. The proposed cryptoprocessor has been synthesized on a Xilinx Vertex II FPGA (xc2v8000) over GF(2^{173}) and it required 159522 clock cycles to perform scalar multiplication and it required 28154 Slices, which is only 60% out of the total number of available Slices.

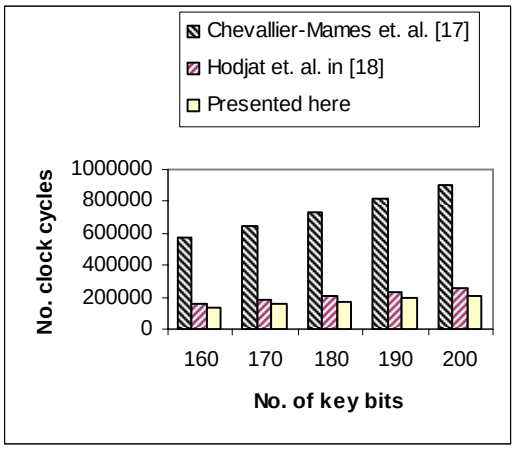

Fig. 5: Time complexities of different designs.

Acknowledgment

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

References

[1] N. Koblitz. Elliptic Curve Cryptosystems, Mathematics of Computation, vol. 48, 1987, pp. 203-209.

[2] A. J. Menezes, "Elliptic Curve Public Key Cryptosystems", Kluwer Academic Publishers, 1993.

[3] ANSI X9.62 - 1998, Public Key Cryptography for the Financial Services Industry: Curve Digital Signature Algorithm (ECDSA), 1998.

[4] IEEE P1363, IEEE Standard Specifications for Public-Key Cryptography, 2000.

[5] National Institute of Standards and Technology, Recommended Elliptic Curves for Federal Government Use, Appendix to FIPS 186-2, 2000.

[6] Standards for Efficient Cryptography Group/Certicom Research, SEC 1: Elliptic Curve Cryptography, Version 1.0, 2000. http://www.secg.org/.

[7] Standards for Efficient Cryptography Group/Certicom Research, SEC 2: Recommended Elliptic Curve Cryptography Domain Parameters, Version 1.0, 2000. http://www.secg.org/.

[8] Wireless Application Protocol (WAP) Forum, Wireless Transport Layer Security (WTLS) Specification. http://www.wapforum.org/.

[9] C. Kocher, "Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems", CRYPTO '96, LNCS 1109, pp. 104-113, 1996.

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

[11] R. Lidl and H. Niederreiter. Introduction to finite fields and their applications. Cambridge University Press, Cambridge, UK, revised edition, 1994.

[12] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson, "Optimal normal bases in GF(p^m)" Discrete Appl. Math., vol. 22, pp. 149–161, 1988/1989.

[13] D. Gordon, "A Survey of Fast Exponentiation Methods", Journal of Algorithms, 1998, pp. 129-146.

[14] M. Joye and C. Tymen, "Compact Encoding of Non-Adjacent Forms with Applications to Elliptic Curve Cryptography", Public Key Cryptography, vol. 1992 of Lecture Notes, in Computer Science, pp. 353-364, Springer-Verlag, 2001.

[15] H. Cohen, T. Ono, and A. Miyaji, "Efficient elliptic curve exponentiation using mixed coordinates", In Advances in Cryptology -SIACRYPT '98 (1998), K. Ohta nd D. Pei, Eds., vol. 1514 of Lecture Notes in Computer Science, pp. 51-65.

[16] J. Omura and J. Massey. Computational method and apparatus for finite field arithmetic. U.S. Patent Number 4,587,627, May 1986.

[17] B. Chevallier-Mames, M. Ciet, M. Joye, "Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity", In IEEE Transactions on Computers, Volume 53 (6), pages 760-768, June 2004.

[18] A. Hodjat, D. Hwang, and I. Verbauwhede, "A scalable and high performance elliptic curve processor with resistance to timing attacks," Proc. IEEE International Conference on Information Technology (ITCC 2005), pp. 538-543, April 2005.

Turki F. Al-Somani received the B.S. and M.S. degrees in Electrical and Computer Engineering from King Abdul-Aziz University in 1997 and 2000, respectively. He is now pursuing his PhD in the area of cryptographic hardware designs at King Fahd University of Petroleum & Minerals.

M. K. Ibrahim received his BSc and PhD degrees from the University of Newcastle Upon Tyne, UK, in 1982 and 1985 respectively. He currently holds the post Chair of Technology, at the School of Engineering and Technology, De Montfort University, UK. His research interests are in computer arithmetic, signal and image processing, audio and video compression, security and cryptosystems, digital rights management, theories of information, application specific processors and FPGAs. He is a senior member of the IEEE and a member of the IEE.