Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (24): 4587-4594, 2008
ISSN 1812-5654

High Performance Elliptic Curve Scalar Multiplication with Resistance Against Power Analysis Attacks

[1]T.F. Al-Somani and [2]A.A. Amin
[1]Department of Computer Engineering, Umm Al-Qura University, Makkah, Saudi Arabia
[2]Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

Abstract: This study presents a high performance scalar multiplication method with resistance against simple and differential power attacks. The scalar multiplier k is divided into a number of partitions that are independently processed in some random order. Each key partition is associated with a precomputed point to keep its significance. Portions of random lengths of randomly selected key partitions are processed depending on a random number that specifies the number of bits to be processed from the randomly picked partition. After processing these randomly selected bits, another portion of another key partition is randomly selected to be processed and so on until all key bits are processed. The curve points resulting from processing the key partitions are assimilated at the end to produce the target scalar product kP. The proposed randomization causes the relation between any leaked information and the actual private key k to be fully confused. The comparison results show that the proposed method outperforms other recent countermeasures in terms of reducing the number of point additions without losing immunity against Simple Power Analysis (SPA), doubling, Refined Power Analysis (RPA) and Zero Value Point (RVP) attacks.

Key words: Elliptic curve cryptosystems, power analysis attacks, simple power attacks, differential power attacks, scalar multiplication

INTRODUCTION

Elliptic Curve Cryptosystems (ECCs) have been recently attracting increased attention (Koblitz, 1987). Standards for ECCs have been adopted by IEEE, ANSI, NIST, SEC and WTLS. The ability to use smaller key sizes and the computationally more efficient ECC algorithms compared to those used in earlier public key cryptosystems such as Rivest *et al.* (1978) and ElGamal (1985) are two main reasons why ECCs are becoming more popular. They are considered particularly suitable for implementation on smart cards or mobile devices. Because of the physical characteristics of such devices and their use in potentially hostile environments, Power Analysis Attacks (PAA) (Kocher, 1996; Kocher *et al.*, 1999) on such devices are considered serious threats. Power analysis attacks seek to break the security of these devices through observing their power consumption trace or computations timing. Careless or naive implementations of cryptosystems may allow power analysis attacks 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.

An elliptic curve E over the finite field GF(p) defined by the parameters a, b $\in$ GF(p) with p > 3, consists of the set of points P = (x, y), where x, y $\in$ GF(p), that satisfy the Equation:

$$y^2 = x^3 + ax + b \quad (1)$$

where, a, b $\in$ GF(p) and $4a^3 + 27b^2 \neq 0 \bmod p$, together with the point at infinity O which is the additive identity of the group (Koblitz, 1987). The number of points #E on an elliptic curve over a finite field GF($q = p^m$) is defined by Hasse's theorem (McEliece, 1987). The set of discrete points on an elliptic curve forms an abelian group, whose group operation is known as point addition. Elliptic curve point addition is defined according to the chord-tangent process. Point addition over GF(p) is described as follows:

Let P and Q be two distinct points on an elliptic curve E defined over GF(p) with Q $\neq$ -P (Q is not the additive inverse of P). The addition of the two points P and Q is the point R (R = P + Q), where R is the additive inverse of S, with S being the third point on E intercepted by the straight line through points P and Q. The additive inverse of a point P = (x, y) $\in$ E is the point -P = (x, -y), which is

Corresponding Author: Turki F. Al-Somani, Department of Computer Engineering, Umm Al-Qura University, P.O. Box 715, Makkah 21955, Saudi Arabia Tel: +9662-5580650 Fax: +9662-5583444

the reflection of the point P with respect to the x-axis on E. When P = Q and P ≠ -P the addition of P and Q is the point R (R = 2P), where R is the additive inverse of S with S being the third point on E intercepted by the straight line tangent to the curve at point P. This operation is referred to as point doubling.

The finite field GF(2^m) is of particular importance in cryptography since it leads to 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 (Lidl and Niederreiter, 1994). 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) ∈ GF(2^m) × GF(2^m) to the Equation:

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

where, a and b ∈ GF(2^m), b ≠ 0, together with the point at infinity.

It is well known that E forms a commutative finite group, with O being the group identity, under the addition operation. Explicit formulas for the addition rule involve several field arithmetic operations (addition, squaring, multiplication and inversion) in the underlying finite field. The group operation in affine coordinate system involves finite field inversion, which is a very costly operation, particularly over prime fields. Projective coordinate systems are used to eliminate the need for performing inversion. Several projective coordinate systems have been proposed in the literature including homogeneous, Jacobian, Chudnovsky-Jacobian, modified Jacobian and mixed coordinate system (Cohen *et al*., 1998).

Adding a point P on the elliptic curve E to itself a number of times (k) is known as the scalar product (kP) of point P by the scalar k. Scalar multiplication is a basic operation for ECCs which, in the group of points of an elliptic curve, is analogous to exponentiation in the multiplicative group of integers modulo a fixed integer m. The scalar multiplication operation (kP) yields a point on the elliptic curve which is the result of adding point P to itself k times. Several scalar multiplication methods have been proposed by Gordon *et al.* (1998). Computing kP can be done using a straightforward binary method, the double-and-add method, based on the binary expression of the multiplier k. Computing kP using the binary method is described as follows:

Let k = (k_{m-1},…,k_0), where k^{m-1} is the most significant bit of k, be the binary representation of k. The multiplier k can be written as:

$$k = \sum_{0 \le i < m} k_i 2^i = k_{m-1}2^{m-1} + k_{m-2}2^{m-2} + \cdots + k_1 2 + k_0 \quad (3)$$

Using the Horner expansion, k can be rewritten as:

$$k = (\ldots((k_{m-1}2 + k_{m-2})2 + \ldots + k_1)2 + k_0) \quad (4)$$

Accordingly, kP can be written as:

$$kP = 2(\ldots 2(2(2k_{m-1}P + k_{m-2}P) + \ldots + k_1P) + k_0P \quad (5)$$

The binary method algorithm is shown below:

Algorithm 1 (Double-and-add)

1: input P, k
2: Q ← P
3: for i from m-2 downto 0 do
   3.1: Q ← 2Q
   3.2: if k_i = 1 then Q ← Q + P
4: end for
5: output Q

The binary scalar multiplication method is the most straightforward scalar multiplication method. It inspects the bits of the scalar multiplier k, if the inspected bit k_i = 0, only point doubling is performed. If, however, the inspected bit k_i = 1, both point doubling and point addition are performed. The binary method requires m point doublings and an average of m/2 point additions.

Power analysis attacks are usually divided into two types. The first type, Simple Power Analysis (SPA) attack, which is based on a single observation of power consumption, while the second type, Differential Power Analysis (DPA) attack combines SPA attack with an error-correcting technique using statistical analysis (Kocher, 1996; Kocher *et al*., 1999). More importantly, classical DPA attacks have been extensively researched for each cryptosystem and new types of DPA attacks are continuously being developed. Many of the existing countermeasures are vulnerable to the more recent attacks including the Doubling Attack (Fouque and Valette, 2003), the Refined Power Analysis (RPA) (Goubin, 2003) and the Zero-Value Point Analysis (ZVP) (Akishita and Takagi, 2003).

SPA attacks consist of observing the power consumption during a single execution of a cryptographic algorithm. The power consumption analysis may also enable one to distinguish between point addition and point doubling in Algorithm 1. Coron (1999) suggested performing point addition and point doubling in each loop iteration where the result of the point addition operation may be either accepted or ignored based on the k_i value (Algorithm 2, known as the double-and-add-always algorithm).

Algorithm 2 (Double-and-add-always)

1: input P, k
2: Q[0]← P
3: for i from m-2 downto 0 do
   3.1: Q[0] ← 2Q[0]
   3.2: Q[1] ← Q[0]+P
   3.3: Q[0] ← Q[k_i]
4: end for
5: output Q[0]

The disadvantage of Algorithm 2 is the added dummy point additions. Algorithm 2 requires (m-1) point doubling and (m-1) point additions. The side-channel atomicity countermeasure proposed by Chevallier-Mames *et al.* (2004) is an efficient countermeasure against SPA attacks which requires no additional computation overhead. In this method computations of point operations are divided into atomic blocks, which are indistinguishable from each other and hence do not leak out any data regarding the type of operation being performed. Even though the proposed countermeasures in Coron (1999) and Chevallier-Mames *et al.* (2004) are resistant to SPA attacks, they remain vulnerable to DPA attacks.

DPA attacks use error correction techniques and statistical analysis to extract small differences in the power consumption signals. Several countermeasures have been proposed to provide security against DPA attacks (Coron, 1999; Ha and Moon, 2002; Okeya and Sakurai, 2000; Liardet and Smart, 2001; Joye and Quisquater, 2001; Joye and Tymen, 2001; Mamiya *et al.*, 2004; Zhang *et al.*, 2007; Ciet and Joye, 2003). These countermeasures include algorithms based on randomizing the private exponent (Coron, 1999), blinding the base point P (Coron, 1999; Ha and Moon, 2002; Mamiya *et al.*, 2004; Zhang *et al.*, 2007; Ciet and Joye, 2003), randomizing the projective coordinates (Coron, 1999), using a random isomorphism of an elliptic curve (Joye and Tymen, 2001) and using special forms of certain elliptic curves (the Montgomery form (Okeya and Sakurai, 2000), the Jacobian form (Liardet and Smart, 2001) and the Hessian form (Joye and Quisquater, 2001)). All of these countermeasures, however, add computational overhead and are still vulnerable to the more recent DPA attacks, e.g., the Doubling attack (Fouque and Valette, 2003), the Refined Power Analysis (RPA) (Goubin, 2003) and the Zero-Value Point (ZVP) attack (Akishita and Takagi, 2003).

The Doubling attack works only for the most-to-least version of the double-and-add multiplication algorithm (Algorithm 1). The doubling attack assumes that an attacker can detect when the same point operation is performed twice. More precisely, if 2A and 2B are computed in any operations, the attacker is not able to guess the value of A or B but he can check if A = B or A ≠ B. This assumption is reasonable since this kind of computation usually takes many clock cycles and largely depends on the value of the operands. With negligible noise, a simple comparison of the two power traces during the doubling operation can detect such equality.

Goubin (2003) proposed a new DPA attack, namely the Refined Power Analysis (RPA) attack. The RPA attack assumes that the attacker can adaptively input selected messages or elliptic curve points to the victim's scalar multiplication algorithm. Smart (2003) has analyzed the RPA attack in detail and has discounted its effectiveness in a large number of orders. However, the RPA attack is still a threat to most elliptic curve cryptosystems.

The Zero-Value Point (ZVP) attack is an extension of the RPA attack. Whereas in the RPA attack, an attacker uses a special point that has a zero-value coordinate, in the ZVP attack, the attacker utilizes an auxiliary register that might take a zero-value in the definition field. The ZVP attack constitutes a serious threat to elliptic curve cryptosystems.

Resistance against these recent DPA attacks can be achieved by combining two or more of the countermeasures proposed in the literature thus far. In order to protect against the doubling attack, randomizing the projective coordinates or a random field isomorphism should be used. To protect against RPA and ZVP attacks, however, the base point P or the scalar multiplier k should be randomized. Hence, to protect against all these recent DPA attacks, for instance, randomizing the scalar multiplier and randomizing the projective coordinates, can be selected together.

To speedup scalar multiplication, precomputations may need to be performed. The cost of such precomputations is tolerable particularly if these precomputations are utilized in more than one session, e.g., when the elliptic curve point is fixed as in the case of generating public keys. For a particular secure communication session using public keys, however, the elliptic curve point changes as it depends on the public key of the communicating entities, i.e., it is session dependant. The same applies for digital signatures. Since is not unlikely that the elliptic curve point is different for each session, the overhead of precomputations must be considered as part of the total computational time. This study describes a high performance scalar multiplication method that uses precomputations and randomizations to secure ECC against SPA and DPA attacks.

THE PROPOSED SCALAR MULTIPLICATION METHOD

The proposed scalar multiplication method is presented here. The main idea is to divide the binary representation of the scalar multiplier k into a number of partitions that are sequentially processed in any randomized order. Each key partition is associated with a precomputed point to keep its significance. Key partitions are not fully processed, but rather portions of random lengths of these partitions are processed at a time. Control jumps from the processing of some portion of one key partition to process a portion of another key partition till all key partitions are fully processed. Thus, key partitions are partially processed depending on a random number that specifies the number of bits to be processed from the randomly selected partition. Processing of each key partition produces a point on the elliptic curve. After all key partitions are fully processed, the resulting curve points are assimilated to produce the scalar product kP. For u partitions, (u-1) points need to be both precomputed and stored. Further, to allow partial processing of key partitions, storage should be allocated for an accumulation point for each partition to hold the current point value for this partition. In addition, a pointer is needed for each key partition to define the bits processed thus far in this particular partition.

The precomputed points are computed using a sequence of doubling operations of the base point P. The performance of this method, however, depends on how frequently these precomputations need to be performed. Highest performance is achieved when the elliptic curve point and the number as well as the sizes of the key partitions are fixed. In this case, precomputations are performed only once thus significantly reducing the computational overhead. Scalar multiplication with varying elliptic curve point or different number or sizes of key partitions needs to perform precomputations whenever a new session is going to be established. For a particular secure communication session using public keys, the elliptic curve point changes as it depends on the public key of the communicating entities, i.e., it is session dependant. The same comment applies when using digital signatures. Accordingly, the precomputations overhead must be considered as part of the total computational time.

The proposed scalar multiplication method splits the scalar multiplier k into a random number (u) of partitions each of randomly picked size. Thus, the u key partitions are $k = (k^{(u-1)} \| k^{(u-2)} \| \ldots \| k^{(1)} \| k^{(0)})$ with corresponding key-partition lengths of $(L^{(u-1)} \| L^{(u-2)} \| \ldots \| L^{(1)} \| L^{(0)})$. Accordingly, the key size m is given by

$$m = \sum_{i=0}^{u-1} L^{(i)}.$$

In this case, the precomputation step requires $(m-L^{(u-1)})$ point doubling operations. Accordingly, higher values of $L^{(u-1)}$ yield lower precomputation overhead. Even though increasing the number of key partitions provides more security, an increased number of key partitions (u) results in more space overhead since the number of precomputed points (u-1) will also increase. Likewise, the delay overhead increases if the number of key partitions is increased since (u-1) extra point additions are required to assimilate the partial results to produce the scalar product kP. Computing kP, accordingly, can be performed as follows:

The u key partitions; $k = (k^{(u-1)} \| k^{(u-2)} \| \ldots \| k^{(1)} \| k^{(0)})$, are associated with a set of precomputed points to keep the significance of each partition. The scalar product kP is computed as follows:

$$\begin{aligned} kP &= (k^{(u-1)} \| k^{(u-2)} \| \ldots \| k^{(1)} \| k^{(0)}).P \\ &= (2^{size(u-1)} k^{(u-1)} + 2^{size(u-2)} k^{(u-2)} + \ldots + 2^{size(1)} k^{(1)} + k^{(0)}) P \\ &= (2^{size(u-1)} P) k^{(u-1)} + (2^{size(u-2)} P) k^{(u-2)} + \ldots + (2^{size(1)} P) k^{(1)} \\ &\quad + (P) k^{(0)}) \\ &= P_{u-1} k^{(u-1)} + P_{u-2} k^{(u-2)} + \ldots + P_1 k^{(1)} + P_0 k^{(0)} \\ &= P_{u-1} k^{(u-1)} + P_{u-2} k^{(u-2)} + \ldots + P_1 k^{(1)} + P k^{(0)} \end{aligned}$$

where, P_i (i = 1, 2, ..., u-1) is the precomputed point associated with ith key partition $k^{(i)}$ and

$$Size\ (j) = \sum_{i=0}^{j-1} (\text{size of key partition } k^{(i)}).$$

Thus, each partition $k^{(i)}$ is associated with a precomputed point P_i forming the pair: $(k^{(i)}, P)$, where, $P = P_0$. Furthermore, each key partition is associated with an accumulation point and an index. The accumulation point is used to hold the cumulative result of the progressive processing of the bits of this key partition. The index, however, is used as a pointer to the key partition bit that to be inspected next.

The pseudocode of the proposed scalar multiplication method with precomputations is given in Algorithm 3. The partitioning of the multiplier k into u partitions is performed in Step 2. The (u-1) precomputed points are computed by repeated doubling operations at Step 4. Each partition $k^{(i)}$ is associated with its corresponding precomputed point (P_i) to keep the significance of each partition (Step 5). Further, Step 5

associates an accumulation point (Q_i) and an index ($index^{(i)}$) with each partition (i). Scalar multiplication starts at Step 6. In Step 6.2.1, a key partition that is not fully inspected is randomly chosen. A random number r is picked to specify the number of bits to be inspected from the chosen key partition (Step 6.2.2). Point doubling is performed in Step 6.2.3.1 on the accumulation point of the selected key partition. The bits of the chosen key partition are inspected in Step 6.2.3.2 to check whether a point addition is required or not. If the inspected bit is equal to one, a point addition is performed between the key partition accumulation point and its precomputed point. When all m-bits of the key are processed, the accumulation points of all (u-1) key partitions are added in the accumulation point R (Step 7) producing kP.

Algorithm 3: the proposed scalar multiplication method

1: input: P, k,
2: Randomly partition the key into u partitions $k = (k^{(u-1)} \| k^{(u-2)} \| \ldots \| k^{(1)} \| k^{(0)})$, with random partition sizes that are not equal in general
3: Initialization: $Q \leftarrow P$, $R \leftarrow O$
4: Precomputations:
  4.1: $P_0 \leftarrow Q$.
  4.2: for i = 1 to u - 1 do
    4.2.1: for j = 0 to $L^{(i-1)}$-1 do
      4.2.1.1: $Q \leftarrow 2Q$
    4.2.2: end for
    4.2.3: $P_i \leftarrow Q$
  4.3: end for
5: Associate each key partition with its corresponding precomputed point, accumulation point and index
  5.1: for i = 0 to u-1 do
    5.1.1: $Q_i \leftarrow P_i$ $k^{(i)}_{L^{(i)}-1}$ -- accumulation point of the partition equals the precomputed point of the partition iff the MSB of $k^{(i)} = 1$
    5.1.2: $index^{(i)} = L^{(i)}-2$
    5.1.3: Associate ($k^{(i)}$, P_i, Q_i and $index^{(i)}$)
  5.2: end for
6: Randomized Scalar Multiplications:
  6.1: I = m-1
  6.2: While $i \geq 0$ do
    6.2.1: Randomly select a key partition $k^{(j)}$ that is not fully inspected, i.e. for which $index^{(j)} \geq 0$
    6.2.2: Randomly pick the number of bits r to be inspected from $k^{(j)}$ such that $0 < r \leq index^{(j)} + 1$
    6.2.3: For s = r-1 downto 0 do
      6.2.3.1: $Q_j \leftarrow 2Q_j$
      6.2.3.2: if $k^{(j)}_{Index^{(j)}} = 1$ then $Q_j \leftarrow Q_j + P_j$
      6.2.3.3: $index^{(j)} = index^{(j)}-1$
      6.2.3.4: i = i –1
    6.2.4: end for
  6.3: end While
7: Assimilate the resulting accumulation points:
  7.1: for i = 0 to u-1 do
    7.1.1: $R \leftarrow R + Q_i$
  7.2: end for
8: output R

SECURITY AND PERFORMANCE ANALYSIS

Although an attacker may be able to distinguish the double and add point operations, the adopted scalar multiplication algorithm which processes randomly sized portions of randomly picked key partitions do not allow the attacker to associate these operations with a specific key bit position. The proposed randomization of the scalar multiplication computation causes the scalar multiplier k and the base point to appear as if they are totally randomized which renders the proposed method secure against SPA and DPA attacks.

The total computation time of the proposed method consists of: (1) the precomputations time overhead, (2) the time taken by the partial scalar multiplications of various key partitions and (3) the time taken to assimilate the resulting (u-1) elliptic curve points into the required scalar product. The precomputations overhead consists of $(m-L^{(u-1)})$ point doublings. The scalar multiplication process, however, requires (m-u) point doublings and an average of (m-u)/2 point additions. Finally, with u key partitions, the curve point assimilation step requires (u-1) point additions. Increasing the number of partitions (u) would increase the level of confusion and accordingly the security level on the one hand, but it would also increase the number of point additions and the storage requirements on the other. For u key partitions, the proposed algorithm needs to store (u-1) precomputed points, u accumulation points and u index pointers.

Assuming an average size for the u^{th} partition of (m/u) bits, the average number of point doubling operations for the precomputation step is (m-m/u). Thus, the average total number of point doublings is (2m-m/u-u). The average number of point additions, however, is (m+u)/2-1. For simplicity, the number of field multiplications is used as an indicator for the time complexity of point operations since other field operations, e.g., addition, subtraction and multiplication by a small constant, are much faster than multiplication. To avoid costly field inversion operations, the use of projective coordinates is assumed. Further, we assume

that the number of field multiplications for each point doubling and point addition operations to be T and αT, respectively with values of α in the range $1 < \alpha \leq 3$ (Cohen *et al*., 1998). Table 1 and 2 compare the number of point operations and the total number of field multiplications needed for the proposed scalar multiplication algorithm to those needed for other recently reported scalar multiplication algorithms with DPA resistant countermeasures. The five selected countermeasures are the Binary Random Initial Point (BRIP) method (Mamiya *et al*., 2004), the Modular Scalar Randomization (MSR) reported by Ciet and Joye (2003) and the three methods presented in by Zhang *et al*. (2007). Our comparison was limited to these recent countermeasures since they cover recent attacks including the doubling, RPA and ZVP attacks. The average total number of field multiplications for our proposed algorithm is [(2+α/2-1/u)m-(u+α-uα/2)]T with a lower bound of [(1+α/2)(m-1)]T when the number of key partitions u = 1 and an upper bound of [(1+α)(m-1)]T in case u = m (each key partition has only 1 bit). Table 3 lists the average total number of field multiplications for several values of u and for α = 1, 2 and 3. Table 4 the same information for the other five recently reported algorithms. It is clear that the proposed method has a better performance compared to the other countermeasures under consideration for lower-to-medium values of u. Using higher values of u would improve the security level at the expense of lowered performance that is comparable to the other algorithms under consideration. It is also worth noting that the overhead of precomputations is included in the above formula, which means that the proposed method will be even more efficient in cases where the same precomputed points are utilized for several sessions.

It should be noted that the earlier analysis is based on assuming an average length for the uth partition ($L^{(u-1)}$ =m/u). The performance of the proposed algorithm can be further improved while maintaining higher values of u and hence higher security by constraining $L^{(u-1)}$ to have higher values, e.g. ≥ m/2 while all other partitions may have arbitrary randomized lengths.

As an example, considering the case where $L^{(u-1)}$ = m/2, the average total number of field multiplications in this case is (1.5+α/2)m-[(u+α-uα/2)]T with a lower bound of [(1.5+α/2)m-2]T in case u = 2 and an upper bound of [(1+α) m – (1+α/2)]T for u = m/2+1 where $L^{(u-1)}$ = m/2 and $L^{(i)}$ = 1 for i = 0, 1, …., (m/2-1). Table 5 lists the average total number of field multiplications in case $L^{(u-1)}$ = m/2 for several values of u with α = 1, 2 and 3. For lower and medium values of u, Table 5 shows a clear improvement in performance which is fairly independent of the value of u.

Table 1: The No. of point operations for the proposed method and other recent scalar multiplication algorithms

Countermeasure	Point operations
BRIP (Mamiya *et al*., 2004)	m (DBL+ADD)
MSR (Ciet and Joye, 2003)	m (DBL+ADD)
Algorithm 5 (Zhang *et al*., 2007)	(m+1) (DBL+ADD)
Algorithm 6 (Zhang *et al*., 2007)	m (DBL+ADD)
Algorithm 7 (Zhang *et al*., 2007)	m (DBL+ADD)
Proposed method	(2m-m/u-u/DBL+((m+u)/2-1) ADD

Table 2: Total No. of field multiplications

Countermeasure	No. of field multiplications
BRIP (Mamiya *et al*., 2004)	m (1+α) T
MSR (Ciet and Joye, 2003)	m (1+α) T
Algorithm 5 (Zhang *et al*., 2007)	(m+1) (1+α) T
Algorithm 6 (Zhang *et al*., 2007)	m (1+α) T
Algorithm 7 (Zhang *et al*., 2007)	m (1+α) T
Proposed method	[(2+α/2-1/u)m-[(u+α-uα/2)]

Table 3: The proposed method average total No. of field multiplications for Several Values of the No. of key partitions u for α = 1, 2 and 3

	No. of field multiplications		
u	α = 1	α = 2	α = 3
1	(1.5m-1.5)T	(2m-2)T	(2.5m-2.5)T
2	(2m-2)T	(2.5m-2)T	(3m-2)T
3	(2.17m-2.5)T	(2.67m-2)T	(3.17m-1.5)T
4	(2.25m-3)T	(2.75m-2)T	(3.25m-1)T
5	(2.3m-3.5)T	(2.8m-2)T	(3.3m-0.5)T
m	(2m-2)T	(3m-3)T	(4m-4)T

Table 4: The Total No. of Field Multiplications for other recent Scalar multiplication algorithms

	No. of field multiplications		
Countermeasure	α = 1	α = 2	α = 3
BRIP (Mamiya *et al*., 2004)	2m T	3m T	4m T
MSR (Ciet and Joye, 2003)	2m T	3m T	4m T
Algorithm 5 (Zhang *et al*., 2007)	(2m+2)T	(3m+3)T	(4m+4)T
Algorithm 6 (Zhang *et al*., 2007)	2m T	3m T	4m T
Algorithm 7 (Zhang *et al*., 2007)	2m T	3m T	4m T

Table 5: The proposed method average total number of field multiplications for several values of the number of key partitions u for α = 1, 2 and 3 with ($L^{(u-1)}$ = m/2)

	No. of field multiplications		
u	α = 1	α = 2	α = 3
2	(2m-2)T	(2.5m-2)T	(3m-2)T
3	(2m-2.5)T	(2.5m- 2)T	(3m-1.5)T
4	(2m-3)T	(2.5m- 2)T	(3m-1)T
5	(2m-3.5)T	(2.5m- 2)T	(3m-0.5)T
m/2+1	(2m-1.5)T	(3m-2)T	(4m-2.5)T

CONCLUSION

The proposed randomized partitioning of the scalar key k and the randomized processing order of the key partitions in addition to the partial processing of randomized sizes of these key partitions causes total confusion regarding any leaked information about the secret key. Even if the value of the key bit under

processing is identified, it is impossible to know the exact position of this bit in the secret key. Likewise the base point would appear as if it is randomized since different precomputed points are used for each partition and the value of the precomputed points depend on the randomized sizes and number of key partitions. This randomized view of the secret key and base point renders our algorithm secure against, SPA and DPA including the RPA and ZVP attacks. Furthermore, the proposed secure scalar multiplication method enjoys higher performance compared to other recently reported methods. The performance can be further improved when the most significant key partition is constrained to have larger sizes, e.g. $\geq m/2$.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support of Umm Al-Qura University (UQU) and the support of King Fahd University of Petroleum and Minerals (KFUPM).

REFERENCES

Akishita, T. and T. Takagi, 2003. Zero-value point attacks on elliptic curve cryptosystem. Information Security Conference (ISC'03), LNCS 2851, October 1-3, Springer-Verlag, pp: 218-233.

Chevallier-Mames, B., M. Ciet and M. Joye, 2004. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. IEEE Trans. Comput., 53: 760-768.

Ciet, M. and M. Joye, 2003. (Virtually) Free randomization technique for elliptic curve cryptography. Proceedings of the ICICS-2003, LNCS, 2836, December 15-18, Springer-Verlag, Berlin, pp: 348-359.

Cohen, H., A. Miyaji and T. Ono, 1998. Efficient elliptic curve exponentiation using mixed coordinates. Advances in Cryptology -ASIACRYPT '98, LNCS 1514, October 18-22, Springer-Verlag, New York, pp: 51-65.

Coron, J., 1999. Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Cryptographic Hardware and Embedded Systems-CHES'99, LNCS 1717, Koc, C.K. and C Parr, (Eds.). Springer-Verlag, ISBN: 3-540-66646-X, pp: 292-302.

ElGamal, T., 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. Advances in cryptology: Proceedings of CRYPTO 84, August 19-22, Springer Verlag, pp: 10-18.

Fouque, P. and F. Valette, 2003. The doubling attack-why upwards is better than downwards. Cryptographic Hardware and Embedded Systems-CHES'03, LNCS 2779, September 7-10, Springer-Verlag, pp: 269-280.

Gordon, D., 1998. A survey of fast exponentiation methods. J. Algorithms, 27: 129-146.

Goubin, L., 2003. A refined power-analysis attack on elliptic curve cryptosystems. Public Key Cryptography- PKC'03, LNCS 2567, January 06-08, Springer-Verlag, pp: 199-210.

Ha, J. and S. Moon, 2002. Randomized signed-scalar multiplication of ECC to resist power attacks. Cryptographic Hardware and Embedded Systems-CHES '02, LNCS 2523, August 13-15, Springer-Verlag, London, UK., pp: 551-563.

Joye, M. and C. Tymen, 2001. Protections against differential analysis for elliptic curve cryptography. Cryptographic Hardware and Embedded Systems-CHES '01, LNCS 2162, May 14-16, Springer-Verlag, London, UK., pp: 377-390.

Joye, M. and J. Quisquater, 2001. Hessian elliptic curves and side-channel attacks. Cryptographic Hardware and Embedded Systems-CHES '01, LNCS 2162, May 14-16, Springer-Verlag, London, UK., pp: 402-410.

Koblitz, N., 1987. Elliptic curve cryptosystems. Math. Comput., 48: 203-209.

Kocher, C., 1996. Timing attacks on implementations of diffe-hellman, RSA, DSS and other systems. CRYPTO '96, LNCS 1109, 1996, Springer-Verlag, pp: 104-113.

Kocher, C., J. Jaffe and B. Jun, 1999. Differential power analysis. CRYPTO '99, LNCS 1666, 1999, Springer-Verlag, pp: 388-397.

Liardet, P. and N. Smart, 2001. Preventing SPA/DPA in ECC systems using the Jacobi form. Cryptographic Hardware and Embedded Systems-CHES '01, LNCS 2162, May 14-16, Springer-Verlag, London, UK., pp: 391-401.

Lidl, R. and H. Niederreiter, 1994. Introduction to Finite Fields and Their Applications. 2nd Edn., Cambridge University Press, Cambridge, UK., ISBN-13: 978-0521460941 .

Mamiya, H., A. Miyaji and H. Morimoto, 2004. Efficient countermeasure against RPA, DPA and SPA. Cryptographic Hardware and Embedded Systems-CHES '04, LNCS 3156, August 11-13, Cambridge MA, pp: 343-356.

McEliece, R., 1987. Finite Fields for Computer Scientists and Engineers. 1st Edn., Kluwer Academic Publishers, Boston, ISBN-13: 978-0898381917 .

Okeya, K. and K. Sakurai, 2000. Power analysis breaks elliptic curve cryptosystems even secure against the timing attack. Advances in Cryptology-INDOCRYPT'00, LNCS 1977, December 10-13, Springer-Verlag, London, UK., pp: 178-190.

Rivest, R., A. Shamir and L. Adleman, 1978. A method for obtaining digital signatures and public key cryptosystems. Commun. ACM, 21: 120-126.

Smart, N., 2003. An analysis Goubin's refined power analysis attack. Proceeding of the Cryptographic Hardware and Embedded Systems-CHES '03, LNCS 2779, September 7-10, Springer-Verlag, pp: 281-290.

Zhang, N., Z. Chen and G. Xiao, 2007. Efficient elliptic curve scalar multiplication algorithms resistant to power analysis. Inform. Sci., 177: 2119-2129.