A New Variant Nevine Maurice Ebied's Key Randomization Counter Measures to Power Analysis Attacks on Elliptic Curve Cryptosystems

E.Kesavulu Reddy[1]

Assistant Professor
Dept.of Computer Science
College of CMIS
S.V.University , Tirupati

V.V.Lakshmi Prasad[2]

Research Scholar
Dept.of Computer Science
College of CMIS
S.V.University, Tirupati

ABSTRACT
It is essential to secure the implementation of cryptosystems in embedded devices against side-channel attacks. Namely, in order to resist differential (DPA) attacks, randomization techniques should be employed to decorrelate the data processed by the device from secret key parts resulting in the value of this data. This peak appears only if the attacker's guess of a bit or a digit of the secret key is correct. The attacker's goal is to retrieve partial or full information about a long-term key that is employed in several ECSM executions. In this paper we proposed secret key to calculate the number of bits or digits processed in an n-bit prime integer for SPA attacks in the execution of elliptic curve scalar multiplication executions.

Key Words: *Elliptic curve cryptography, Simple power Analysis, Differential Power analysis*

1. Elliptic Curve Cryptosystems and Side-Channel Attacks

1.1.Introduction

Elliptic curve cryptosystems (ECCs) are suitable for implementation on devices with limited memory and computational capability such as smart cards and also with limited power such as wireless handheld devices. This is due to the fact that elliptic curves over large finite fields provide the same security level as other cryptosystems such as RSA for much smaller key sizes.

Considering power analysis attacks, there are two main types that were presented by Kocher et al. These are simple and differential power analysis attacks (referred to as SPA and DPA respectively).

Both of them are based on monitoring the power consumption of a cryptographic token while executing an algorithm that manipulates the secret key. The traces of the measured power are then analyzed to obtain significant information about the key. In some cases the key can be totally compromised and in others the search space of the key can be reduced to a computationally affordable size. In SPA, a single power trace can reveal large features of the algorithm being executed such as the iterations of the loop. Moreover, cryptosystem-specific operations such as point doubling and adding in ECCs can be identified [3]. In order to resist this SPA attack, the steps of the algorithm need to be uniform across different executions.

Hence, DPA attacks are, in general, more powerful than the SPA attack. Randomization of the data processed at some instant is essential in resisting this type of attacks. Electromagnetic emanations present another powerful side channel since the information is leaked from the device via more than one channel and is a function of space as well as of time. In [2], Agrawal et al. presented both simple (SEMA) and differential (DEMA) electromagnetic analysis attacks on smart cards and on a Palm pilot in [21]. In [AARR], they conclude that software countermeasures rely on signal information reduction, which is achieved by "randomization and/or frequent key refreshing within the computation", which agrees with the concept of resisting DPA attacks.

1.2. Elliptic Curve Cryptosystems

Let K be a finite field and E be an elliptic curve (EC) over K defined by the following Weierstrass equation

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (2.1)$$

where $a_i \in K$ and $\Delta \neq 0$, where Δ is the discriminant of E and is defined in [13 Section 3.1].

Let L be an extension field of K. Then E(L) denotes the set of L-rational points (x, y) on E, where (x, y) 2 L × L and satisfy (2.1), together with the point at infinity ∂. The addition of two points on the curve is performed using a chord-and-tangent rule. E(L) and this

Manuscript received February 5, 2009
Manuscript revised February 20, 2009

addition operation form an abelian group where ∂ is the identity. The point addition operation consists of finite field operations carried in the underlying field K. We denote the field inversion by I, the multiplication by M, the squaring by S. The point addition is denoted by A. When the two operands of the addition are the same point, the operation is referred to as point doubling and is denoted by D.

1.2.1 Elliptic curves over prime fields

If K = Fp, where p > 3 is a prime, (2.1) can be simplified to 1

$$E : y^2 = x3 + ax + b, \qquad (2.2)$$

where a and b ∈ Fp. The discriminant of this curve is Δ = −16 (4a3+27b2). The negative of a point P = (x, y) is −P = (x,−y) such that P + (−P) = ∂. This simplification is generally applicable when the characteristic of K is not 2 or 3.

The affine coordinate (A) representation of a point P = (x, y) can be replaced by projective coordinates representations in order to render the point addition and doubling operations less costly in terms of field operations. The following representations are the best known

• standard (homogeneous) projective coordinates (P); the projective point (X : Y : Z), Z ≠ 0, corresponds to the affine point (X/Z, Y/Z), ∂ corresponds to (0 : 1 : 0) and the negative of (X : Y : Z) is (X : −Y : Z). • Jacobian projective coordinates (J); the projective point (X : Y : Z), Z 6= 0, corresponds to the affine point (X/Z2, Y/Z3), O corresponds to (0 : 1 : 0) and the negative of (X : Y : Z) is (X : −Y : Z).

• Chudnovsky coordinates (C); the Jacobian point (X : Y : Z) is represented as $(X : Y : Z : Z^2 : Z^3)$.

1.2.2 Elliptic curves over binary fields

If K = F ∈ m, (2.1) can be simplified to

$$E : y^2 = x3 + ax + b,, \qquad (2.3)$$

where a and b ∈ F_2 m The discriminant of this curve is Δ = b and the negative of a point P = (x, y) is −P = (x, x + y). Such a curve is known as non-super singular.

Standard and Jacobian projective coordinates are used to represent points on this type of curves in the same way as on the prime curves with the difference that the negative of (X : Y : Z) is (X : X + Y : Z).

1.2.3 Elliptic Curve Scalar Multiplication

Scalar multiplication in the group of points of an elliptic curve is analogous to exponentiation in the multiplicative group of integers modulo a fixed integer. Thus, it is the fundamental operation in EC-based cryptographic systems. The scalar multiplication, denoted kP, is the result of adding the point P to itself k times, where k is a positive integer, that is kP = P + P + · · · + P | {z } k copies and −kP = k(−P). u is said to be the order of P if u is the smallest integer such that uP = ∂.

In many applications, the scalar k is a short-term (ephemeral) or long-term (private) secret (key). From now on, we will always assume that k is a n-bit integer, where n is the bit length of u, the order of the group of points of interest in an ECC. Also the point P may be fixed (e.g., the base point of the ECC) or unknown a priori.

Let $(k_{n-1}, k_{n-2}, \ldots, k_1, k0)_2$ 2 be the binary representation of k,

i.e., $k_i \in \{0, 1\}$ for $0 \le i < n - 1$. Thus,

$$kP = \left(\sum_{i=0}^{n-1} k_i 2^i\right)P$$

$$= 2(2(\cdots 2(2(k_{n-1} P) + k_{n-2} P) + \cdots) + k_1 P) + k_0 P \quad (2.4)$$

$$= (k_{n-1} 2^{n-1} P) + \cdots (k_1 2P) + (k_0 P) \quad (2.5)$$

Hence, kP can be computed using the straightforward double-and-add approach in n iterations. In fact, there are two algorithms that can be used. Algorithm 2.1, which corresponds to the expansion in (2.4), scans the bits of the scalar k from left to right, i.e., from the most significant bit to the least significant one. Algorithm 2.2, corresponding to (2.5), scans the bits of k from right to left. These algorithms are analogous to the square-and-multiply algorithms employed in exponentiation-based cryptosystems.

Algorithm 1.1. Left-to-Right Double-and- Add Algorithm

Input: $k = (k_{n-1} \ldots k_0)$ 2 and P ∈ $E(F_q)$.

Output: KP.

1. Q ← ∂
2. for i from n − 1 down to 0 do
2.1 Q ← 2Q.
2.2 if $(k_i = 1)$ then
Q ← Q + P.
3. Return (Q).

Algorithm 1.2. Right-to-Left Double-and- Add Algorithm

Input: $k = (k_{n-1}, k_0)$2 and P ∈ (F_q).

Output: KP.

1. Q← ∂; R← P.
2. for i from 0 to n − 1 do
2.1 if (k_i = 1) then
Q ← Q + R.
2.2 R ← 2R.
3. Return(Q).

The expected number of point addition (A) and point doubling (D) operations performed in the binary algorithm (left-to-right or right-to-left) is

$$(n-1)\ D + \frac{n}{2}\ A.$$

If affine coordinates are used, the field operation count is 2.5n S + 3n M + 1.5n I. Algorithm 2.1 is usually preferred since one of the addition operands is the base point P which is constant through the algorithm. This has the advantage of saving a register if P is a fixed point known a priori. Moreover, it allows the use of mixed coordinates addition. That is, when one of the operands to the addition operation is fixed, the Z-coordinate of that operand is set to and remains 1, this reduces the number of field multiplications needed to perform the point addition as illustrated in Table 2.1 and Table 2.2.

Hence if Q is stored in Jacobian coordinates and P in affine coordinates, Jacobian coordinates can be used for doubling in Algorithm 2.1 and mixed Jacobian-affine for the Elliptic Curve Cryptosystems and Side-Channel Attacks addition. The field operation count is then 8n M + 5.5n S + (1 I + 3 M + 1 S), where the last three terms are needed to convert the resulting point back to affine coordinates.

1.2.4.Non-Adjacent form (NAF)

The key k can be represented in Non-adjacent Form (NAF) k' = (k'$_n$, . . . , k_1, k_0)2, where $k_i \in \{0,\pm 1\}$ and no two consecutive digits are non zero; that is, $k_{i+1}\, k'_i = 0$ for i ≥ 0 [Rei60; Sol00]. The NAF of an integer is unique and is at most one digit longer than its binary representation. The average density of nonzero digits among all NAFs of length n is n/3.

This key representation requires the slight modification of the binary algorithm that is to subtract, rather than add, P when $k'_i = -1$, i.e., to add −P. This is advantageous for ECCs since the negative of a point can be obtained with a minor cost as we mentioned in Section 2.1, e.g., a modular negation for curves over prime fields and a bit-wise XOR operation for curves over binary fields.

1.2.5. Window methods

This method is sometimes referred to as m−ary method. There are different versions of window methods [17;26]. What is common among them is that, if the window width is w, some multiples of the point P up to $(2^w - 1)$P are precomputed and stored and k is processed w bits at a time. k is recoded to the radix 2^w. k can be recoded in a way so that the average density of the nonzero digits in the recoding is $1/(w + \xi)$, where $0 \le \xi \le 2$ depends on the algorithm. Let the number of precomputed points be t, in the precomputation stage, each point requires either a doubling or an addition to be computed also depending on the algorithm.

This ECSM method is suitable for unknown or fixed point P. The cost is Storage: t points, where $2^{w-2} \le t \le 2^{w-1}$ depending on the algorithm.
Precomputation: t point operations (A or D).
Expected running time:

$$(n-1)\ D + n\frac{n}{w+\xi}\ A,$$

where $0 \le \xi \le 2$ depending on the algorithm. Note that the number of doubling is between n − w and n − 1.

1.2.6.Simultaneous multiple point multiplication

This method is used to compute kP +lS where P may be a known point. This algorithm was referred to as Shamir's trick in [10]. If k and l are n-bit integers, then their binary representations are written in a 2 × n matrix called the exponent array. Given width w, the values iP +jS are calculated for $0 \le i, j < 2^w$. Now the algorithm performs $d = \lceil n/w \rceil$ iterations. In every iteration, the accumulator point is doubled w times and the current 2 × w window over the exponent array determines the precomputed point that is to be added to the accumulator.

Algorithm 1.3. Simultaneous multiple point multiplication (Shamir-Strauss method)
Input: Window width w, $d = \lceil n/w \rceil$,
$k = (K_{d-1}, \ldots, K_1, K_0)_{2^w}$ $l = (L_{d-1}, .., L_1, L_0)_{2^w}$, and P, S ∈ E($F_q$). 3Also according to [Ber01], it is originally due to Straus [Str64].

Output: kP + lS.

1. Precomputation. Compute iP + jS for all i, j ∈ [0, $2^w - 1$].
2. Q ← K_{d-1} P + L_{d-1} S.
3. for i from d − 2 down to 0 do
3.1 Q ← 2^w Q.
3.2 Q ← Q + (K_i P + L_i S).
4. Return(Q).

Storage: $2^{2w} - 1$ points. For w = 1, 3 points.
For w = 2, 15 points.
Precomputation:
($2^{2(w-1)} - 2^{w-1}$) D + ($3 \cdot 2^{2(w-1)} - 2^{w-1} - 1$) A.
For w = 1, 1 A.
For w = 2, 1 D + 11 A.

Expected running time:
(d − 1)w D + $\frac{(2^{2w}-1)}{2^{2w}}$ d − 1) A.

For w = 1, (n − 1) D +($\frac{3}{4}$ n − 1) A.

For w = 2, (n − 1) D +($\frac{15}{32}$ n -1) A.

Using sliding windows can save about 1 4 of the precomputed points and decrease the number of additions to $\frac{n}{w+(1/3)}$, which is about 9% saving for w ∈ {2, 3}.

Interleaving method

This method is also a multiple point multiplication method, that is we want to compute $\sum k^j P_j$ for points P_j and integers k^j. In the comb and simultaneous multiplication methods, each of the precomputed values is a sum of the multiples of the input points. In the interleaving method, each precomputed value is simply a multiple of one of the input points. Hence, the required storage and the number of point additions at the precomputation phase is decreased at the expense of the number of point additions in the main loop. This method is flexible in that each k_j can have a different representation, e.g., different window size, as if a separate execution of a window method is performed for each k^j P_j with the doubling step performed jointly on a common accumulator, as shown in [12]. As an illustration, we provide the following algorithm that computes kP + lS where both k and l are represented to the same base 2^w.

1.2. 7Algorithm 1.4. Interleaving method

Input: width w, d = $\lceil n/w \rceil$,
k = $(K_{d-1}, \ldots, K_1, K_0)_{2^w}$ l= $(L_{d-1}, \ldots, L_1, L_0)_{2^w}$, and P, S ∈ E($F_q$).
Output: kP + lS.

1. Precomputation. Compute iP and iS for all i ∈ [0, $2^w - 1$].
2. Q ← K_{d-1} P.
3. Q ← Q L_{d-1} S.
4. for i from d − 2 down to 0 do
4.1 Q ← 2^w Q.
4.2 Q ← Q + K_i P.
4.3 Q ← Q + L_i S.
5. Return(Q).

Storage: $2^{w+1} - 2$ points.
Precomputation: 2(w-1) D + 2(2^w − w − 1) A.
Expected running time: w(d − 1) D + (2d − 1). $\frac{(2^{2w}-1)}{2^{2w}}$ A

In general, if different basis and/or representations are used for k and l, we have
Storage: 2t points, where $2^{w-2} \le t \le 2^{w-1}$ depending on the particular window algorithm used as discussed in Section 2.1.3.
Precomputation: 2t point operations (A or D).
Expected running time: (n − 1) D + 2 $\frac{n}{w+i}$ A, where 1 $\le i \le 2$ depending on the algorithm

2. Koblitz curves

Koblitz curves [15]—originally named anomalous binary curves—are the curves E_a, a ∈ {0, 1}, defined over
F_2 Ea: $y^2 + xy = x^3 + ax^2 + 1$, (2.6)
which is a special case of (2.3) where b = 1. E_a (F_2^m) is the group of F_2^m-rational points on E_a. Let μ = $(-1)^{1-a}$ that is μ ∈ {−1, 1}. The order of the group is computed as
Ea (F_2^m) = $2^m + 1 - V_m$, (2.7)
where {Vh} is the Lucas sequence defined by
V_0 = 2, V_1 = μ and V $h+1$
= μV_h − 2V $h-1$ for h ≥ 1.

The value of m is chosen to be a prime number so that # Ea (F_2^m) = h · u is very nearly prime, that is u >

2 is prime and h = 3 − μ. The main advantage of Koblitz curves when used in public-key cryptography is that scalar multiplication of the points in the main subgroup, the group of order u, can be performed without the use of point doubling operations . This is due to the following property. Since these curves are defined over F_2^m then if P = (x, y) is a point on E_a, then the point (x^2, y^2) is on the curve, as well. That is the Frobenius (squaring, in this case) endomorphism _ : $E_a(F_2 m) \rightarrow E_a(F_2 m)$ defined by

$(x, y) \mapsto (x^2, y^2)$, $\partial \mapsto \partial$

is well defined. It can also be verified by point addition on E_a tha

$(x^4, y^4) + 2(x, y) = \mu \cdot (x^2, y^2)$.

Hence, the squaring map can be considered as a multiplication by the complex number τ

Satisfying $\tau^2 + 2 = \mu\tau$, (2.8)

that is $\tau = \frac{1}{2}(\mu + \sqrt{-7})$.

The norm of τ is 2. Thus, it is beneficial to represent the key k as an element of the ring $Z[\tau]$,

$$K = \sum_{i=0}^{l-1} k_i \tau^i \qquad (2.9)$$

for some l. We can therefore carry the scalar multiplication kP of a point P on E_a more efficiently by replacing the doubling operation in the double-an-add algorithm by the squaring map. In [25], Solinas has shown how to represent k as in (2.9) in its τ -adic non adjacent form (τ NAF) where $k_i \in \{-1, 0, 1\}$ and $k_{i+1} k_i = 0$ for $i \geq 0$abusing the notation, we will refer to k_i as a sbit. However, this results in $l \approx 2m$. Therefore, he proposed a reduced τ -adic non adjacent form (RTNAF) for k where k is reduced modulo $\delta = (\tau^{m-1})/(\tau - 1)$, hence l = m+ a. He has proved that in a τ NAF representation the number of 0s is $\frac{2}{3}$ on average. He also mentioned that 1 and -1 are equally likely on average.

3. Power and Electromagnetic Analysis Attacks on ECCs

The elliptic curve scalar multiplication Q = kP, where both P and Q are points on the curve and k is an integer, is the fundamental computation performed in ECCs. Usually both P and Q are public information and k is the secret key stored securely in he cryptosystem. The security of the system lies in the difficulty of extracting k from P and Q, which is the hard problem known as EC discrete logarithm problem (ECDLP).

However, the mathematically proved security of a cryptosystem does not imply its implementation security against side-channel attacks. Among those attacks are those that monitor the power consumption and or the electromagnetic emanations of a device, e.g., a smart card or a handheld device, and can infer important information about the instructions being executed or the operands being manipulated at a specific instant of interest.

These attacks are broadly divided into two categories; simple and differential analysis attacks. We will refer to the former category as SPA attacks and the latter as DPA attacks. Though SPA and DPA are the acronyms for simple power analysis and differential power analysis, respectively, are used in this thesis to include simple and differential electromagnetic analysis as well due to their extensive usage in the literature. Also, in subsequent discussions, we may only focus on power analysis attacks, since the countermeasures that we are interested in were proposed to prevent information leakage on side-channels, the power consumption and the electromagnetic emanations.

Power analysis attacks use the fact that the instantaneous power consumption of a hardware device is related to the instantaneous computed instructions and the manipulated data. The attacker could measure the power consumption during the execution of a cryptographic algorithm, store the waveform using a digital oscilloscope and process the information to learn the secret key. Kocher et al., in [16], first introduced this type of attack on smart cards performing the DES operation. Then Messerges et al. [19] augmented Kocher's work by providing further analysis and detailed examples of actual attacks they mounted on smart cards.

In general, SPA attacks are those based on retrieving valuable information about the secret key from a single leaked information—power consumption or electromagnetic emanation—trace. On the other hand, DPA attacks generally include all attacks that require more than one such trace along with some statistical analysis tools to extract the implicit information from those traces.

3.1 SPA Attack on ECCs and its Countermeasures

Coron [3] has transferred the power analysis attacks to ECCs and has shown that an unaware implementation of EC operations can easily be exploited to mount an SPA attack. Monitoring of the power consumption enables us to visually identify large features of an ECC implementation such as the main loop in Algorithms 2.2 and 2.1. Moreover, it may also enable to

recognize the exact instruction that has been executed. For example, if the difference in power consumption between point doubling (D) and point addition (A) is obvious in their respective power traces, then, by investigating one power trace of a complete execution of a double-and-add algorithm, the bits of the scalar k are revealed. That is, whenever a D is followed by A, the corresponding bit is $k_i = 1$, otherwise if D is followed by another D, then $k_i = 0$. This sequence of point operations is referred to as the DA sequence.

Window methods process the key on a digit (window) level. The basic version of this method, that is where _ = 0 in Section 2.1.3, is inherently uniform since in most iterations, w D operations are followed by 1 A, except for possibly when the digit is 0. Therefore, fixed-sequence window methods were proposed [20;23;27] in order to recode the digits of the key such that the digit set does not include 0.

3.2 DPA Attack on ECCs and its Countermeasures

When the relation between the instructions executed by a cryptographic algorithm and the key bits is not directly observable from the power signal, an attacker can apply differential power analysis (DPA). DPA attacks are in general more threatening and more powerful than SPA attacks because the attacker does not need to know as many details about how the algorithm was implemented. The technique also gains strength by using statistical analysis and digital signal processing techniques on a large number of power consumption signals to reduce noise and to amplify the differential signal. The latter is indicated by a peak, if any, in the plot of the processed data. This peak appears only if the attacker's guess of a bit or a digit of the secret key is correct. The attacker's goal is to retrieve partial or full information about a long-term key that is employed in several ECSM executions.

As for the SPA attack, Kocher et al. were the first to introduce the DPA attack on a smart card implementation of DES [16]. Techniques to strengthen the attack and a theoretical basis for it were presented by Messerges et al. in [18;3,19]. Coron applied the DPA attack to ECCs [3].

In order to resist DPA attacks, it is important to randomize the value of the long-term key involved in the ECSM across the different executions. Some of the countermeasures that were based on randomizing the key representation [22;12] were proven to be inadequate since the intermediate point computed in the accumulator Q at certain iteration remained one of two possible values [11]. The constancy of the value of this intermediate point is an integral part in the success of first-order DPA attacks.

A potential DPA countermeasure is known as key splitting [14]. It is based on randomly splitting the key into two parts such that each part is different in every ECSM execu+tion. An additive splitting using subtraction is attributed to Clavier and Joye [CJ01]4. It is based on computing

$kP = (k - r)P + rP$, (I)

The authors mention that the idea of splitting the data was abstracted in [5]. where r is a n-bit random integer, that is, of the same bit length as k. Alternatively, Ciet and Joye [8] suggest the following additive splitting using division, that is, k is written as

$k = \lfloor k/r \rfloor + (k \bmod r)$. (1)

Hence, if we let $k_1 = (k \bmod r)$, $k_2 2 = \lfloor k/r \rfloor$ and S = rP, we can compute $KP = k_1 p + k_2 P$ (II)

where the bit length of r is n/2. They also suggest that (II) should be evaluated with Shamir-Strauss method as in Algorithm 2.3. However, they did not mention whether the same algorithm should be used to evaluate (I). The following multiplicative splitting was proposed by Trichina and Bellezza [0] where r is a random integer invertible modulo u, the order of P. The scalar multiplication kP is then evaluated as

$kP = [kr^{-1} \pmod u](rp)$ (III)

To evaluate (III), two scalar multiplications are needed; first R = rP is computed, then kr^{-1} R is computed.

4. Key Splitting Methods

4.1. Introduction

We discuss different the forms of key splitting along with their strengths and weaknesses. We also discuss the candidate SPA-resistant algorithms and compare the resulting performance when combined with each form of key splitting. At the end of the chapter, we present countermeasures to DPA attacks on the ECDSA and the ECMQV algorithms.

This approach was suggested by Clavier and Joye in [CJ01] and revisited by Ciet [Cie03] as follows. In order to compute the point kP, the n-bit key k is written as

$k = k_1 + k_2$,

such that $k_1 = k - r$ and $k_2 = r$, where r is a random integer of length n bits. Then kP is computed as

$kP = k_1 P + k_2 P$. (5.1)

It is important to note that each of the terms of (5.1) should be evaluated separately and their results combined at the end using point addition. That is the multiple-point multiplication methods that use a common accumulator to save doubling operations such as

Algorithms 2.3 and 2.4—whether at the bit level (w ≡ 1) or window level (w > 1)-should not be used, even when a countermeasure against SPA is employed. This observation is based on the following lemma. Let $k_{b\to a}$ denote $\lfloor k(\mathrm{mod}\,2^{b+1})2^{a} \rfloor$ or, simply, the bits of k from bit position b down to bit position a, with $b \geq a$.

Lemma 4.2 Let splitting scheme I in (6.1) be evaluated using Algorithm 2.3 with w = 1 (d = n). Then, at the end of some iteration j, 0 < j ≤ n − 1, there are only two possible values for Q, those are $[k_{n-I-j}]$ P or $[k_{n-I-j} - 1]$ P.

Proof. Algorithm 2.3—and similarly Algorithm 2.4—computes the required point by scanning

$k_1 = (k_{1\ n-1}, \ldots, k_{1\ 0})2$ and

$k_2 = (k_{2\ n-1}, \ldots, k_{2\ 0})_2$ from the most significant end down to the least significant end. Hence, at the end of iteration j, the accumulator Q contains the value

$$Q = k_{1n-I-j}\ P + k_{2n-I-j}\ P \qquad (5.2)$$

$$= [k_{1n-I-j} + k_{2n-I-j}]\ P.$$

We can write k, k_1 and k_2 as

$$k = k_{n-I-j^{j}}\,2^{j} + k_{j-1-0} \qquad k_i =$$

$$k_{i\ n-I-j^{j}}\,2^{j} + k_{i\ j-1-0} \qquad (5.4)$$

Since $k = k_1 + k_2$ we have

$$k_{1\ j-I-0} + k_{2\ j-I-0}\,j = k_{j-I-0} + b\,2^{j} \text{ where } b \in \{0, 1\} \qquad (5.5)$$

and

$$k_{1\ n-I\to j} + k_{2\ n-I\to j} = k_{n-I\to j} - b$$

The DPA attack would proceed in the same way, whether the algorithm processes a single bit or a digit per iteration, though it would be more involved in the latter case depending on the digit size. The attacker can double the number of traces gathered and compute the necessary intermediate points as if there was no countermeasure in place.

4.3 Modular Division:

n the following algorithm, a and b are integers internally represented each by an array of w-bit digits. The length of each array is $d = \lceil n/w \rceil$ digits. Note that for the modular inversion, as mentioned by Savas and Ko¸c [25], b needs not be less than the modulus u, but be in $[1, 2^{m}-1]$, where m = dw. Also note that the values R^2 mod u, where $R = 2^{m}$, and u' are computed once per modulus, i.e., per curve.

Algorithm 4.4. Modular division

Input: u: a n-bit prime, $d = \lceil n/w \rceil$, m = dw, $R^2 \pmod u = (2m)^2 \pmod u$, $u' = u^{-1} \bmod 2^{w}$, $a \in [1, p-1]$ and $b \in [1, 2^{m}-1]$.

Output: $ab^{-1} \pmod u$.

1. Compute $b^{-1}R \pmod u$ using Algorithm 6.6.
2. Compute $x = a(b^{-1}R)R^{-1} \pmod u$ using Algorithm 6.5.
3. Return(x).

The following algorithm is Algorithm 14.36 in [21]. We include it here for the sake of completeness.

Algorithm 4.5Montgomery multiplication [21]

Input: u: a n-bit prime, $d = \lceil n/w \rceil$, m = dw, $u' = u^{-1} \bmod 2^{w}$, $x = (x_{d-1}\ldots, x0)2^{w}$ and

$y = (y_{d-1}\ldots y0)2^{w}$.

Output: $xy2^{-m} \pmod u$.

1. A ← 0. // A = (ad, $a_{d-1}, \ldots,$ a0)2w
2. for i from 0 to d − 1 do
2.1 ui ← $(a_0 + x_i y_0) \bmod 2^{w}$
2.2 A ← $(A + x_i y_0 + u_i m)/2^{w}$
3. if (A > u) then
A A ← u.
4. Return(A).

The following algorithm was presented by Savas and Ko¸c in [25] as the modified Kaliski-Montgomery Inverse.

Algorithm 4.6. Montgomery inversion

Input: u: a n-bit prime, $d = \lceil n/w \rceil$, m = dw, $R^2 \pmod u = (2^{m})^2 \pmod u$, $u' = u^{-1} \bmod 2^{w}$ and $b \in [1, 2^{m}-1]$.

Output: $b^{-1}R \pmod u$.

1. Compute f and $x = b^{-1}2f \pmod u$ using Algorithm 6.7, Where $n \leq f \leq m + n$.
2. if ($f \leq m$) then
2.1 x ← $xR^2R^{-1} \pmod u$ using Algorithm 6.5. // $x = b^{-1}2^{m+f} \pmod u$
2.2 f f + m. // f > m, $x = b{-}12f \pmod u$
3. x ← $x2^{2m-f}R^{-1} \pmod u$ using Algorithm 6.5.
// $x = b^{-1}2^{f}\ 2^{2m-f}$
$2^{-m} = b^{-1}2^{m} \pmod u$
4. Return(x).

4.7. Existing System:

Nevine Maurice Ebied's modified the almost Montgomery inverse algorithm of [ScKK00] to be resistant to SPA attacks as in the following algorithm. SwapAddress(c, d) denotes interchanging the memory addresses of the integer's c and d. This is an inexpensive operation, hence its usage as a dummy operation to balance the branches of the main loop. We implemented the "if" statement in steps 3.4 and 3.5 such that the number of conditions checked per loop iteration is always three. In assembly language, this can be easily ensured. Written in Java, step 3.4 is implemented as
if((xLSb = = 0) && (xLSb = = 0) &&(xLSb = = 0)).
If the condition is false, due to short-circuit evaluation, the flow control will move to the following "if" after the first check, otherwise, it will perform the check three times. The following "if"—step 3.5—is similar but with the condition checked only two times
if((yLSb = = 0) && (yLSb = = 0)).

Algorithm 4.7. Almost Montgomery inverse

Input: u: a n-bit prime,
$d = \lceil n/w \rceil$, m = dw and b $\in$ [1, $2^m - 1$].

Output: f and $b^{-1} 2^f \pmod u$, where $n \le f \le m + n$.
1. x ← u; y ← b; r ← 0; s ← 1.
2. f ← 0.
3. while (v > 0) do
3.1 U ← x − y; V ← −U.
3.2 T ← r + s.
3.3 f ← f + 1.
3.4 if (((lsb(x) = 0))) then //
This "if" is special
SwapAddress(x, U); SwapAddress
(x, U) // dummy
SHR(x); SHL(s).
3.5 else if ((lsb(y) = 0)) then //
This "if" is special
SwapAddress(y, V); SwapAddress
(y, V) // dummy
SHR(y); SHL(r).
3.6 else if (V >= 0) then
SwapAddress(y, V);
SwapAddress(s, T)
SHR(y); SHL(r).
3.7 else
SwapAddress(x, U);
SwapAddress(r, T)
SHR(x); SHL(s).
4. T ← u − r; V ← u + T .
5. if (T > 0) then
Return(f, T)
else
Return(f, V).

The drawback of this algorithm is that an SPA of the number of iterations of the main loop directly leaks the value of f. If f is uniformly distributed, the search space of b is reduced from 2^w to $2^{m-\log 2^m}$, which is not a significant reduction. It is interesting to study how f is actually distributed.

4.8.Proposed System

We modified **the Nevine Maurice Ebied's Almost Montgomery inverse** and **A NEW VARIANT** of [ScKK00] to be resistant to SPA attacks as in the following algorithm.
Linear Congruence's

A congruence of the form

$$ax \equiv b \pmod m$$

where m is a positive integer, a and b are integers , and x is a variable, is called Linear congruence . Such congruences arise throughout number theory and its applications.

Definition : If a and b are integers, then a is said to be **congruent** to b modulo n, write a ≡ b (mod n), if n divides (a – b). The integer n is called the modulus of the congruence.

Definition : The **equivalence class** modulo n of an integer b is the set of all integers congruent to b modulo n.

Definition : The ring of **integers modulo** n, denoted by Z_n , is the set of (equivalence classes of) is the integers {0,1,2,, n–1}. Addition, subtraction, and multiplication in Z_n are performed modulo n.

Algorithm 4.9: Modified Montgomery Inversion

Input: u: a n-bit prime, $d = \lceil n/w \rceil$,
m = dw, $R^2 \pmod u = (2^m)^2 \pmod u$, $u' = u^{-1} \bmod 2^w$ and b $\in$ [1, $2^m - 1$].
t: No of precomputed points $1 \le t \le n$
w: Window width Least significant of bit 2^{w-z}
$t \le 2^w - 1$
Output: $b^{-1} R \pmod u$.

1. Select a number b such that $(b, 2^m) = 1$
2. Compute b^{-1} such that $bb^{-1} \equiv 1 \pmod{2^m}$
3. If f > m then $x = b^{-1} 2^f \pmod u$ ∵ $x = b^{-1} 2^f \pmod u$

4. If f $\leq$ m then
5. x $\leftarrow R^2 R^{-1} (\mathrm{mod}\, u)$ $\because$ R = 2^m
6. x = b $^{-1}$ 2^{m+f} (mod u)
   f $\leftarrow$ m+f
7. Return(x)

In the modified Montgomery Inverse Algorithm of Savas and Koc , we select f such that gcd $(b, 2^m) = 1$, m $\leq$ f $\leq$ m +n . So b is not reduced from 2^m to $2^{m-\log 2^m}$. Therefore this is significant reduction and hence f is not uniformly distributed and it can't leaks the value

5. Conclusion

We modified **the Nevine Maurice Ebied's Almost Montgomery inverse** and **A New Variant** of [ScKK00] **of Montgomery Inversion** i.e **Modified Montgomery Algorithm** to be resistant to SPA attacks . Also the proposed algorithm of Modified Montgomery Algorithm eliminate an SPA of the number of iterations of the main loop directly leaks the value of f and f is uniformly distributed with a significant reduction.

Reference

[1] Nevine Maurice Ebied's Key Randomization Counter Measures To Power Analysis Attacks On Elliptic Curve Cryptosystems Ph.D. thesis, University of Waterloo, Ontario, Canada, 2007

[2] D. Agrawal, B. Archambeault, J. R. Rao & P. Rohatgi. The EM Side- Channel(s): Attacks and Assessment Methodologies. Internet Security Group, IBM Watson Research Center.ps. 2, 3

[3] J.-S. Coron. "Resistance against differential power analysis for elliptic curve cryptosystems". In Cryptographic Hardware and Embedded Systems – CHES '99, LNCS, vol. 1717, pp. 292–302. Springer-Verlag, 1999. 2, 22, 24, 158, 170, 180, 181, 186

[4] M. Ciet. Aspects of Fast and Secure Arithmetics for Elliptic Curve Cryp- tography. Ph.D. thesis, Universit´e Catholique de Louvain, 2003. 120, 170, 171, 180

[5] S. Chari, C. S. Jutla, J. R. Rao & P. Rohatgi. "Towards sound approaches to counteract power-analysis attacks." In Advances in Cryptology – CRYPTO '99, LNCS, vol. 1666, pp. 398–412. Springer-Verlag, 1999. 24

[6] [Cie03] M. Ciet. Aspects of Fast and Secure Arithmetics for Elliptic Curve Cryp- tography. Ph.D. thesis, Universit´e Catholique de Louvain, 2003. 120, 170, 171, 180

[7] C. Clavier & M. Joye. "Universal exponentiation algorithm a first step towards provable SPA-resistance". In Cryptographic Hardware and Embedded Systems – CHES '01, LNCS, vol. 2162, pp. 300–308. Springer-Verlag, 2001. 4, 24, 120

[8] M. Ciet & M. Joye. "(Virtually) free randomization techniques for elliptic curve cryptography". In Information and Communications Security – ICICS '03, LNCS, vol. 2836, pp. 348–359. Springer-Verlag, 2003. 4, 25, 122, 164, 181

[9] M. Ciet, J.-J. Quisquater & F. Sica. "Preventing differential analysis in GLV elliptic curve scalar multiplication". In Cryptographic Hardware and Embedded Systems – CHES '02, LNCS, vol. 2523, pp. 540–550. Springer- Verlag, 2003. 4, 25, 126, 173, 179

[10] T. ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". IEEE Transactions on Information Theory, 31(4):469– 472, 1985.

[11] P.-A. Fouque, F. Muller, G. Poupard & F. Valette. "Defeating countermeasures based on randomized BSD representations". In Cryptographic Hardware and Embedded Systems – CHES '04, LNCS, vol. 3156, pp. 312–327. Springer-Verlag, 2004. 4, 24, 76, 101, 174

[12] J. Ha & S. Moon. "Randomized signed-scalar multiplication of ECC to resist power attacks". In Cryptographic Hardware and Embedded Systems – CHES '02, LNCS, vol. 2523, pp. 551–563. Springer-Verlag, 2002. 3, 24, 27, 31, 40, 58, 70, 75, 93, 101, 173

[13] D. Hankerson, A. Menezes & S. Vanstone. Guide to Elliptic Curve Cryptography. Springer-Verlag, 2004. 8, 9, 10, 11, 18, 130, 131, 138, 145, 168, 197 [HP] C. Heuberger & H. Prodinger. Personal communication. August, 2003. 210 [HVZ02] V. C. Hamacher, Z. G. Vranesic & S. G. Zaky. Computer Organization. Boston: McGraw-Hill, fifth ed., 2002. 136

[14] M.Joy " Defenes against side channel analysis". In I.F.Blake G. Seroussi & N.P. smart, editors, Advances in Elliptic Curve Cryptography, chap5. Cambridge University Press, 2005. 2, 24.

[15] N. Koblitz. "CM curves with good cryptographic properties". In Advances in Cryptology – CRYPTO '91, LNCS, vol. 576, pp. 279–287. Springer-Verlag, 1992. 19, 8

[16] P. Kocher, J. Jaffe & B. Jun. "Differential power analysis". In Advances in Cryptology – CRYPTO '99, LNCS, vol. 1666. Springer-Verlag, 1999. 2, 22, 24, 172, 194

[17] A. Miyaji, T. Ono & H. Cohen. "Efficient elliptic curve exponentiation". In Information and Communication Security, First International Conference – CICS '97, LNCS, vol. 1334, pp. 282–

290. Springer-Verlag, 1997. 15 [Mon87] P.-L. Montgomery. “Speeding the Pollard and elliptic curve methods of factorization”. Mathematics of Computation, 48:243–264, 1987. 23, 131, 161

[18] T. S. Messerges, E. A. Dabbish & R. H. Sloan. “Investigations of power analysis attacks on smart cards”. In USENIX Workshop on Smart- card Technology, pp. 151–161. May 1999. 2, 24, 172

[19] [T. S. Messerges, E. A. Dabbish & R. H. Sloan. “Examining smart card security under the threat of power analysis attacks”. IEEE Transactions on Computers, 51(5):541–552, May 2002. 22, 24

[20] B. M¨oller. “Securing elliptic curve point multiplication against sidechannel attacks”. In International Security Conference – ISC ’01, LNCS, vol. 2200, pp. 324–334. Springer-Verlag, 2001. Extended version.pdf. 23, 164,196

[21] A. J. Menezes, P. C. van Oorschot & S. A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996. 139

[22] E. Oswald & M. Aigner. “Randomized addition-subtraction chains as a ountermeasure against power attacks”. In Cryptographic Hardware and Embedded

[23] Systems – CHES ’01, LNCS, vol. 2162, pp. 39–50. Springer-Verlag, 2001. 3, 24, 27, 28, 70, 173

[24] K. Okeya & T. Takagi. “The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks”. In Topics in Cryptology – CT-RSA ’03, LNCS, vol. 2612, pp. 328– 343. Springer-Verlag, 2003. 23, 128, 164, 197

[25] P. Rohatgi, D. Agrawal, B. Archambeault, S. Chari & J. R. Rao. “Power, EM and all that: Is your crypto device really secure?” A talk given as part of the 7th Workshop on Elliptic Curve Cryptography – ECC 2003, 2003. 3 G. W. Reitwiesner. “Binary arithmetic”. Advances in Computers, 1:231– 308, 1960. 14, 15, 31, 51, 58, 70

[26] E. Savas & C¸ etin Kaya Koc¸. “The Montgomery modular inverserevisited.” IEEE Transactions on Computers, 49(7):763–766, 2000. 138, 139, 140

[27] J. A. Solinas. “Efficient arithmetic on Koblitz curves”. Designs, Codes and Cryptography, 19:195–249, 2000. 4, 14, 15, 20, 37, 46, 50, 58, 70, 89, 90, 91, 92, 95, 110, 112

[28] N. Th´eriault. “SPA resistant left-to-right integer recodings”. In Selected Areas in Cryptography – SAC ’05, LNCS, vol. 3897, pp. 345–358. Springer-Verlag, 2006. 23, 128, 133, 164

Authors Biography

I am E.kesavulu Reddy working as Assistant Professor in Dept. of. Computer Science (MCA) SVU College of CMIS, Tirupati (AP)and also worked as a Head in Dept of Computer Applications at SiTech Tirupati (AP) in India. I have six years experience in teaching and three years in the area of Cryptography and Network Security. I obtained MCA degree with First Class from SV University and M.Phil 2nd Class from Madurai Kamaraj University, Madurai. Now I am doing the PhD in Part-Time in Dept.of.Computer Science under the guidance of Prof. P. Govinda Rajulu in Dept.of Computer Science
SV University Tirupati