Elliptic Curve Cryptosystems and Side-channel Attacks

Ekambaram Kesavulu Reddy
Department of Computer Science, Sri Venkateswara University
E. Kesavulu Reddy., P.N. Kalva, Thukivakam, Renigunta, Andhra Pradesh, India (Email:ekreddy2002@yahoo.com)
(Received July 24, 2009; revised and accepted Nov. 5, 2009)

Abstract

In this paper, we present a background on elliptic curve cryptosystems (ECCs) along with the different methods used to compute the scalar multiplication (ECSM), which is the core operation of ECCs, and the various costs associated with them. We have also provided a brief background on Simple (SPA) and Differential (DPA) power and electromagnetic analysis attacks on the classical ECSM algorithms. We study on minor collisions and to provide an analytic result for their probability of occurrence as well as effect of the fixed sequence window method.

Keywords: Differential power analysis, elliptic curve cryptosystems, side-channel attacks, simple power analysis

1 Introduction

In this paper, we present an overview of elliptic curve cryptosystems (ECCs) [2, 23]. The primary operation of ECCs is the elliptic curve scalar multiplication (ECSM) [26]. Hence we discuss about original ECSM algorithms are susceptible to side-channel analysis (SCA) attacks.

1.1 Elliptic Curve Cryptosystems

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

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \qquad (1)$$

where $a \times K$ and $\triangle \neq 0$, where $\triangle$ is the discriminant of E.

Let L be an extension field of K. Then $E(L)$ denotes the set of L-rational points (x, y) on E, where $(x, y) \times L \times L$ and satisfy Equation (1), together with the point at infinity ϑ. The addition of two points on the curve is performed using a chord-and-tangent rule. $E(L)$ and this addition operation form an Abelian group where ϑ 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 Elliptic Curves over Prime Fields

If $K = F_p$, where $p > 3$ is a prime, Equation (1) can be simplified to

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

where a and $b \times F$, The discriminant of this curve is $\Delta = -16(4a^3 + 27b^2)$. The negative of a point $P = (x, y)$ is $-P = (x, -y)$ such that $P + (-P) = \vartheta$. 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 o standard (homogeneous) projective coordinates (P):

The projective point $(X : Y : Z)$, $Z \neq 0$, corresponds to the affine point $(X/Z, Y/Z)$, ϑ 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 $(X/Z_2, Y/Z_3)$, 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 $(X : Y : Z : Z^2 : Z^3)$.

1.3 Elliptic Curves over Binary Fields

If $K = F_{2^m}$, Equation (1) can be simplified to Equation (2). Where a and $b \times F_{2^m}$.The discriminant of this curve is $\delta = b$ and the negative of a point $P = (x, y)$ is $-P = (x, x + y)$. Such a curve is known as non-supersingular. 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 $(X : X + Y : Z)$.

1.4 Elliptic Curve Scalar Multiplication (ECSM)

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 $kP = P + P + \cdots + P \,|\{z\}k$ copies and $-kP = k(-P)$. u is said to be the order of P if u is the smallest integer.

Let $(k_{n-1}, k_{n-2}, \cdots, k_1, k_0)_2$ be the binary representation of k, i.e., $k \times \{0, 1\}$ for $0 \leq i < n-1$. Thus,

$$\begin{aligned} kP &= (\sum_{i=0}^{n-1} 2^i)P \\ &= 2(2(\cdots 2(2(k_{n-1}P) + k_{n-2}P) + \cdots) \\ &\qquad + k_1P) + k_0P \\ &= (k_{n-1}2^{n-1}P) + \cdots (k_1 2P) + (k_0 P). \end{aligned}$$

Hence, kP can be computed using the straightforward double-and-add approach in n iterations (see Algorithms 1 and 2). These algorithms are analogous to the square-and-multiply algorithms employed in exponentiation-based cryptosystems.

Algorithm 1 Left-to-Right Double-and-Add Algorithm

1: Input: $k = (k_{n-1} \cdots k_{k_0})$ 2 and $P \in E(F_q)$.
2: Output: kP.
3: $Q \leftarrow \eth$
4: **for** i from $n-1$ down to 0 **do**
5: $\quad Q \leftarrow 2Q$.
6: $\quad$ **if** $(k = 1)$ **then**
7: $\quad\quad Q \leftarrow Q + P$.
8: $\quad$ **end if**
9: **end for**
10: Return (Q).

Algorithm 2 Right-to-Left Double-and- Add Algorithm

1: Input: $k = (k_{n-1}, \cdots, k_{k_0})$ 2 and $P \in E(F_q)$.
2: Output: KP.1.$Q \leftarrow \eth$; $R \leftarrow P$.
3: $2(2(\cdots 2(2(k_{n-1}P) + k_{n-2}P) + \cdots) + k_1P) + k_0P$ $= (k_{n-1}2^{n-1}P) + \cdots (k_1 2P) + (k_0 P)$
4: for i from 0 to n?1 do
5: **if** if $(k_i = 1)$ then **then**
6: $\quad Q \leftarrow Q + R$.
7: **end if**
8: $R \leftarrow 2R$.
9: Return (Q).
10: End

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

2 Window Methods

This method is sometimes referred to as m-ary method. What is common among them is that, if the window width is w, some multiples of the point P up to $(2^w\text{-}1)P$ 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 $0 \leq \xi \leq 2$ 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 $2^{w-2} \leq t \leq 2^{w-1}$ depending on the algorithm.

Precomputation: t point operations (A or D).

Expected running time: $(n-1)\ D + n\frac{n}{w+\xi}A$, where $0 \leq \xi \leq 2$ depending on the algorithm.

2.1 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 [9] (see Algorithm 3). If k and l are n-bit integers, then their binary representations are written in a $2 \times n$ matrix called the exponent array. Given width w, the values $iP + jS$ are calculated for $0 \leq i, j < 2^w$. Now the algorithm performs $d = \lceil n/w \rceil$ iterations. In every iteration, the accumulator point is doubled w times and the current $2 \times w$ window over the exponent array determines the precomputed point that is to be added to the accumulator.

Algorithm 3 Simultaneous multiple point multiplication (Shamir-Strauss method)

1: Input: Window width w, $d = \lceil n/w \rceil$, $k = (K_{d-1}, \cdots, K_1, K_0)_{x^w}$ $l = (L_{d-1}, \cdots, L_1, L_0)_{2^w}$, and $P, S \in E(F_q)$
2: Output: $kP + 1S$.
3: Precomputation. Compute $iP + jS$ for all $i, j \in [0, 2^w\text{-}1]$.
4: $Q \leftarrow K_{d-1}P + L_{d-1}S$.
5: **for** i from $d-2$ down to 0 **do**
6: $\quad Q \leftarrow 2^wQ$.
7: $\quad 2Q \leftarrow Q + (K_iP + L_iS)$.
8: **end for**
9: Return (Q).

Storage: $2^{2w} - 1$ points. For $w = 1, 3$ points. For $w = 2, 15$ points.

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

Expected running time: $(d-1)wD + \frac{2^{2w-1}}{2^{2w}}d - 1)A$.
For $w = 1$, $(n-1)D + (n-1)A$.
For $w = 2$, $(n-1)D + (\frac{15}{32}n - 1)A$.

Using sliding windows can save about 14 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}.

2.2 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 integer's k^j (see Algorithm 4). 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.

Algorithm 4 Interleaving method

1: Input: Window width w, $d = \lceil n/w \rceil$, $k = (K_{d-1}, \cdots, K_1, K_0)_{x^w}$ $l = (L_{d-1}, \cdots, L_1, L_0)_{2^w}$, and

$$P, S \in E(F_q)$$

2: Output: $kP + 1S$.
3: Precomputation. Compute iP and iS for all $i \in [0, 2^w - 1]$.
4: $Q \leftarrow K_{d-1}P$.
5: **for** i from $d-2$ down to 0 **do**
6: $\quad Q \leftarrow 2^w Q$.
7: $\quad Q \leftarrow Q + K_i P$
8: $\quad Q \leftarrow Q + L_i S$
9: **end for**
10: Return (Q).

Storage: $2^{2w} + 1$ 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 $2^{w-2} \leq t \leq 2^{w-1}$;

Precomputation: $2t$ point operations (A or D);

Expected running time: $(n-1)D + 2\frac{n}{w+1})A$, where $1 \leq i \leq 2$ depending on the algorithm.

3 Power and Electromagnetic Analysis Attacks on ECCS

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 handhold 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.

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 [15], first introduced this type of attack on smart cards performing the DES operation. Then Messerges et al. [17] 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 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 [7] 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 3 and 4.

Window methods process the key on a digit (window) level. The basic version of this method, that is where ϑ = 0 in Section 2.1, is inherently uniform since in most iterations, wD operations are followed by 1A, except for possibly when the digit is 0. Therefore, fixed-sequence window methods were proposed [19, 21, 25] 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 [15]. Techniques to strengthen the attack and a theoretical basis for it were presented by Messerges et al. in [16, 17]. Coron applied the DPA attack to ECCs [7]. 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 execution. An additive splitting using subtraction is attributed to [6]. It is based on computing

$$kP = (k-r)P + rP. \quad (3)$$

The authors mention that the idea of splitting the data was abstracted in [1]. where r is a n-bit random integer, that is, of the same bit length as k. Alternatively, [4] suggest the following additive splitting using division, that is, k is written as

$$k = \lfloor k/r \rfloor + k \bmod r.$$

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

$$KP = k_1 P + k_2 P, \quad (4)$$

where the bit length of r is $n/2$. They also suggest that Equation (4) should be evaluated with Shamir-Strauss method as in Algorithm 2.1. However, they did not mention whether the same algorithm should be used to evaluate Equations 3.

The following multiplicative splitting was proposed by Trichina and Bellezza [5] where r is a random integer invertible modulo u, the order of P. The scalar multiplication kP is then evaluated as

$$kP = [kr^{-1} (\mathrm{mod} u)](rp).$$

To evaluate the above equation, two scalar multiplications are needed; first $R = rP$ is computed, and then $kr^{-1}R$ is computed.

4 Key Splitting Methods

4.1 Additive Splitting Using Subtraction (Scheme I)

This approach was suggested by [6] and revisited by [3] 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

$$kP = k_1 P + k_2 P. \quad (5)$$

It is important to note that each of the terms of Equation (3) should be evaluated separately and their results combined at the end using point addition. This observation is based on the following lemma. Let $k_{b \to a}$ denotes $\lfloor k(\bmod 2^{b+1}) 2^a \rfloor or$, simply, the bits of k from bit position b down to bit position a, with $b \geq a$.

Lemma 1. *Let splitting scheme I can be evaluated using Algorithm 2.2 with $w = 1 (d = n)$. Then, at the end of some iteration j, $0 < j \leq n-1$, there are only two possible values for Q, those are $[k_{n-I-j}]P$ or $[k_{n-I-j} - 1]P$.*

Proof. Algorithm 2.2 and similarly Algorithm 2.1 computes the required point by scanning $k = (k_{1_{n-1}}, \cdots, k_{1_0})_2$ and $k_2 = (k_{2_{n-1}}, \cdots, k_{1_0})_2$ from the most significant end down to the least significant end. Hence, at the end of iteration j, the accumulator Q contains the value

$$\begin{aligned} Q &= k_{1n-I-j} P + k_{2n-I-j} P \\ &= [k_{1n-I-j} + k_{2n-I-j}]P. \end{aligned}$$

We can write k, k_1 and k_2 as

$$\begin{aligned} k &= k_{2n-I-j^j} 2^j + k_{j-1-0}. \\ k &= k_{1_{j-I-1}} 2^j + k_{i_{j-1-0}}. \end{aligned}$$

Since $k = k_1 + k_2$ we have

$$k_{1j-I-0} + k_{2_{j-I-0}} j = k_{2_{j-I-0}} j + b2^j,$$

where $b \in \{0, 1\}$, and $k_{1_{n-I-j}} + k_{2_{n-I-j}} = k_{n-I \to j} - b$.

The 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.

Hence each term of Equation (5) should be computed separately using a SPA-resistant algorithm fixed-sequence window method. If the key splitting process, e.g., the subtraction in this case, is processing the key every time, then an attacker can obtain less noisy information about the key words such as their Hamming weights by averaging the side-channel trace obtained from the key splitting process.

Moreover, if it is difficult for the attacker to locate the instances where the key is manipulated, then by correlating different traces, he can detect where the same data is processed. Therefore, it is desirable to use a previously split version of the key to generate the new one. Hence in Equation (5), k_1 and k_2 can be refreshed as $k_2 \pm rt$, $k_2\ \mu\ rt$, before the t-th execution of the ECSM, where the addition /subtraction is modulo the group order of the points on the elliptic curve and r_t is an n-bit random integer. □

4.2 Additive Splitting Using Division (Scheme II)

As an alternative to the previous splitting in [4] suggest that a random divisor r be chosen and the key k written as $k = g * r + h$, where $g = \lfloor k/r \rfloor$ and $h = k \bmod r$. Let $S = rP$, then kP can be computed as

$$kP = gS + hP. \tag{6}$$

We choose the bit length of r to be $l = \lceil n/2 \rceil$. That is, r is chosen uniformly at random from the range $[2^{l-1}, 2^l - 1]$. Hence, the bit length of g is at most $\lfloor n/2 \rfloor + 1 \leq l + 1$ and at least l and that of h is at most l [13].

An ECSM is first performed to compute the point S, where the scalar is of size half that of k. Then, unlike splitting scheme I, a multiple point multiplication method can be safely used. In the following, we will justify this assertion.

Let the representations of k, g and h to the base 2^w, for some $w \geq 1$, be $(K_{2z-1}, \cdots, K_1 1, K_0)2^w$, $(G_z, \cdots, G1, G0)2^w$ and $(H_{z-1}, \cdots, H_1, H_0)$, respectively, where $z = \lceil l/w \rceil$ that is $l \leq zw\_l + w \leq 1$. If $l < zw$, then $G_z = 0$, otherwise, if $l = zw$, then $Gz \leq 1$. As before, let $K_{b \to a}$ denotes $\lfloor k \bmod 2^{bw+1}/2^{awe} \rfloor$ or, simply, the w-bit digits of k from digit position b down to digit position a, with $b \geq a$. Let Equation (6) be evaluated using Algorithms 2.1 or 2.2, replacing d by $z + 1$ in these algorithms and setting $H_z = 0$. Then at the end of some iteration j, $1 < j\_z$, the accumulator Q contains the value

$$\begin{aligned} Q &= G_{z \to j} S + H_{z \to j} P, \\ &= (G_{z \to j} * r + H_{z \to j}) P. \end{aligned}$$

Let $k^j = G_{z \to j} * r + H_{z \to j}$, which is of length $2z - j$ w-bit digits1. In general- exceptions follow-, $k \neq K_{2z-1 \to j}$. This is true since $K_{2z-1 \to j} = G_{z \to j} * r + h_j$, where h_j is the l-bit remainder of the division of $K_{2z-1 \to j}$ by r. Since $K_{2z-1 \to j} \neq k$, then from the division theorem, the pair $(G_{z \to j}, h_j)$ is not equal to (g, h), hence, in general $h_j \neq H_{z \to j}$.

4.2.1 Major Collisions

A major collision is defined as the occurrence of $k^j = K_{2z-1 \to j}$ at some iteration $j \in [1, z\text{ - }1]$. The intermediate point computed at this value of k^j is the same value that would be computed when no countermeasure is in place. The condition of this collision is provided by the following lemma.

Lemma 2. *For some $j \in [1, z\text{ - }1]$, $k^j = K_{2z-1 \to j} 2z - 1$ if $G_{j-1 \to 0} = 0$.*

Proof. We have

$$\begin{aligned} k &= g * r + h \\ &= (G_{z \to j} * 2^{jw} + G_{j-1 \to 0}) * r \\ &\qquad + (Hz_{z \to j} * 2^{jw} + H_{j-1 \to 0}) \\ &= k^j * 2^{jw} + G_{j-1 \to 0} * r + H_{j-1 \to 0}. \end{aligned}$$

But $k = K_{2z-1 \to j} * 2^{jw} + K_{2z-1 \to 0}$. Hence, if $Gj_{j-1 \to 0} = 0$, we have $k^j = K_{2z-1 \to 0}$ and $K_{j-1 \to 0} = H_{j-1 \to 0}$. On the other hand, if $k^j = K_{2z-1 \to j}$, then $b(!G_{j-1 \to 0} * r)/2^{jw} = 0$. However, $r \geq 2^{l-1}$ that is, $r \geq 2^{(z-1)w}$. Hence, $G_{j-1 \to 0} = 0^2$.

The probability of the occurrence of this collision is around 2^{-jw}. That is, it increases with the iterations of a multiple-point multiplication ECSM algorithm. It is negligible in the first iterations that are critical for the attacker in a DPA attack. Moreover, these collisions can be avoided when evaluating Equation (12) for all j as follows. After performing the division of k by r, the quotient g is inspected. If the least significant w bits are found to be 0, another r is chosen. Note that this incurs a negligible reduction in the choice space of r from 2^{l-1} to approximately $2^{l-1}2^{l-w-1}$.

Another way to avoid these collisions is to make the quotient g always odd. That is if g is even, it is decremented by one and h is updated by adding r to it. This may increase the bit length of h to $l + 1$. □

4.2.2 Minor Collisions

A minor collision occurs when at some iteration $j \in [1, z]$, for two values of r: r_1 and r_2, such that $r_1 \neq r_2$, we have $k_1^j = k_2^j j \neq K_{2z-1 \to j}$. The conditions favoring these collisions are not straightforward to analyze. Some of them occur when $h1 = h2$, but also many of them occur with $g \neq g$ and $h_1 \neq h_2$. Also in some cases, collisions occur when $gcd(r_1, r_2) \neq 1$, where gcd is the greatest common divisor.

In the following we refer to ϕ as a t-time collision value, if at iteration j, $k^j = \phi$ for t different values of r. We have conducted some experiments to study the probability of happening of these collisions for $n = 40$ and 50 when divided by all divisors of length 20 and 25 bits, respectively, with window width $w = 4$. We found that for different values of j, after excluding the values of g with w least significant bits, about 63% of the values of k^j on average were collision-free. About 25.6% were two-times collision values. The maximum number of collisions t for some value varied with the iteration; it was higher towards the middle iterations than the first and last iterations. For example, in the middle iterations, some 40-bit integers exhibited k^j values with up to 132-times collision and up to 1735-times for 50-bit integers. The density of values

that have the higher number of collisions is usually 1 or 2. On the other hand, after the first iteration, the maximum number of collisions we obtained was 12 for 40-bit integers and 23 for 50-bit integers.

5 Proposed System

5.1 Major Collisions

A Major collisions is defined as the occurrence of $k^j = k_{2z-1} \rightarrow j$ at some iteration $j \in [1, z-1]$. The intermediate point computed of this value of k^j is the same value that would be computed when no counter measure is in place.

Lemma 3. *For some j ∈ [1, z - 1], $k^j = K_{2z-1\rightarrow j} \rightarrow j$ iff $G_{j-1\rightarrow 0} = 0$. We know that*

$$\begin{aligned} K &= g * r + h, \\ g &= G_{z\rightarrow j} * 2^{jw} + G_{j-1\rightarrow 0} \\ h &= H_{z\rightarrow j} * 2^{jw} + H_{j-1\rightarrow 0}. \end{aligned}$$

Now

$$\begin{aligned} K &= (G_{z\rightarrow j} * 2^{jw} + G_{j-1\rightarrow 0} * r) \\ &\quad +(H_{z\rightarrow j} * 2^{jw} + H_{j-1\rightarrow 0}) \\ &= (G_{z\rightarrow j} * 2^{jw}) * r + G_{j-1\rightarrow 0} * r \\ &\quad +(H_{z\rightarrow j} * 2^{jw} + H_{j-1\rightarrow 0} \\ K &= (G_{z\rightarrow j} * r + H_{j-1\rightarrow 0}) * 2^{jw} \\ &\quad +G_{j-1\rightarrow 0} * r + H_{j-1\rightarrow 0}. \end{aligned}$$

Let

$$\begin{aligned} k^j &= G_{z\rightarrow j} * r + H_{z\rightarrow j} \\ K &= k^j * 2^{jw} + G_{j-1\rightarrow 0} * r + H_{j-1\rightarrow 0}. \end{aligned} \quad (7)$$

Case 1:
Let us assume that $G_{j-1\rightarrow 0} = 0$. From Equation (7), we have

$$K = k^j * 2^{jw} + H_{j-1\rightarrow 0}. \quad (8)$$

But we know that

$$k^j = k_{2z-1} \rightarrow j * 2^{jw} + k_{j-1\rightarrow 0}. \quad (9)$$

From Equations (8) & (9), $k^j = k_{2z-1} \rightarrow j$ and $H_{j-1\rightarrow 0} = k_{j-1\rightarrow 0}$.

Case 2:
Let us assume that $K^j = K_{2z-1} \rightarrow j = 0$. From Equation (7), we have

$$K = k_{2z-1} \rightarrow j * 2^{jw} + G_{j-1\rightarrow 0} * r + H_{j-1\rightarrow 0}. \quad (10)$$

But we know that

$$K = k_{2z-1} \rightarrow j * 2^{jw} + k_{j-1\rightarrow 0}. \quad (11)$$

From Equations (10) & (11), we have

$$\begin{aligned} G_{j-1\rightarrow 0} + H_{j-1\rightarrow 0} &= k_{j-1\rightarrow 0}\frac{G_{j-1\rightarrow 0} * r}{2^{jw}} + \frac{H_{j-1\rightarrow 0}}{2^{jw}} \\ &= \frac{k_{j-1\rightarrow 0}}{2^{jw}} \\ &\quad \lfloor\frac{G_{j-1\rightarrow 0} * r}{2^{jw}}\rfloor + \lfloor\frac{H_{j-1\rightarrow 0}}{2^{jw}}\rfloor \\ &= \lfloor\frac{K_{j-1\rightarrow 0}}{2^{jw}}\rfloor + \lfloor\frac{G_{j-1\rightarrow 0} * r}{2^{jw}}\rfloor + 0 \\ &= 0 \\ \therefore \lfloor\frac{G_{j-1\rightarrow 0} * r}{2^{jw}}\rfloor &= 0 \\ r \geq 2^{l-1} &\Rightarrow r \geq 2^{(z-1)w} \\ z - 1 &= \lceil l/w \rceil \nrightarrow^{r}_{2^{jw}} 0 \\ w(z-1) &\leq l - 1 \\ \therefore G_{j-1\rightarrow 0} &= 0. \end{aligned}$$

The Probability of major collisions $= \frac{2^{(z-1)w/2^{jw}}}{2^{(z-1)w}} = \frac{1}{2^{jw}} = 2^{-jw}$.

5.2 Minor Collisions

A Minor collision occurs when at some iteration $j \in [1, z]$ for two values of r: r_1 and r_2, such that $r_1 \neq r_2$ we have $k_1^j = k_2^j \neq k_{2z-1} \rightarrow j$.

Lemma 4. *Probability of the occurrence of the minor collision is around $\frac{2^{-jw}}{2}$.*

Proof. We know that $k = g * r + h$ and $k_1^j = G_{1z\rightarrow j} * r_1 + H_{z\rightarrow j}$. Now $k_1^j = G_{2z\rightarrow j} * r_2 * r_2 + H_{2z\rightarrow j}$ and $k_2^j = G_{2z\rightarrow j} * r_2 + H_{2z\rightarrow j}$. □

Case 1:
Let

$$h_1 = h_2 \Rightarrow H_{1z\rightarrow j} = H_{2z\rightarrow j}.$$

For $k_1^j = k_2^j$, we have

$$\begin{aligned} G_{1z\rightarrow j} * r_1 + H_{1z\rightarrow j} &= G_{2z\rightarrow j} * r_2 + H_{2z\rightarrow j} \\ G_{1z\rightarrow j} * r_1 &= G_{2z\rightarrow j} * r_2. \end{aligned}$$

From the above equations,

$$\lfloor\frac{G_{1z\rightarrow j} * r_1}{2^{1+jw}}\rfloor = \lfloor\frac{G_{2z\rightarrow j} * r}{2^{1+jw}}\rfloor. \quad (12)$$

Since

$$\begin{aligned} z &= \lceil\frac{l}{w}\rceil \\ z &\leq \frac{l-1}{w} \\ 1 + zw &\leq 1 \\ r_1 \geq 2^l \quad &\& \quad r_2 \geq 2^l \\ r_1 \geq 2^{1+jw} \quad &\& \quad r_2 \geq 2^{1+jw} \\ \frac{r_1}{2^{1+jw}} \geq \frac{2^{1+zw}}{2^{1+jw}} \quad &\& \quad \frac{r_2}{2^{1+jw}} \geq \frac{2^{1+zw}}{2^{1+jw}}. \end{aligned}$$

Therefore $\frac{r_2}{2^{1+jw}} \nrightarrow 0$ and $\frac{r_2}{2^{1+jw}} \nrightarrow 0$. It is existed in Equation (12) only when $G_{1z\rightarrow j} = 0$ and $G_{2z\rightarrow j} = 0$. Therefore, the probability of occurrence of minor collision: $\frac{2^{1+zw}/2^{1+jw}}{2^{1+zw}} = \frac{1}{2^{1+jw}} = \frac{2^{-jw}}{2}$. The above probability is the half of that of major collisions.

Case 2:
Let $g_1 \neq g_2$ and $h_1 \neq h_2$. For $K_1^j = K_2^j$,

$$\begin{aligned} G_{1z} * r_1 + H_{1z\rightarrow j} * r_2 &= G_{2z\rightarrow j} * r_2 + H_{2z\rightarrow j} \\ +\lfloor \frac{H_{1z\rightarrow j}}{2^{1+jw}} \rfloor &= \lfloor \frac{G_{2z\rightarrow j} * r}{2^{1+jw}} \rfloor. \end{aligned}$$

Since

$$\begin{aligned} z &= \lceil \frac{l}{w} \rceil, \\ z &\leq \frac{l-1}{w}, \\ 1 + zw &\leq 1. \end{aligned}$$

Now

$$\begin{aligned} r_1 \geq 2^l \quad &\text{and} \quad r_2 \geq 2^l. \\ r_1 \geq 2^{1+jw} \quad &\text{and} \quad r_2 \geq 2^{1+jw}. \\ \frac{r_1}{2^{1+jw}} \geq \frac{2^{1+zw}}{2^{1+jw}} \quad &\text{and} \quad \frac{r_2}{2^{1+jw}} \geq \frac{2^{1+zw}}{2^{1+jw}}. \\ \frac{r_1}{2^{1+jw}} \nrightarrow 0 \quad &\text{and} \quad \frac{r_2}{2^{1+jw}} \nrightarrow 0. \end{aligned}$$

Equation (12) possible existed only when $G_{1z\rightarrow j} = 0$ and $G_{2z\rightarrow j} = 0$. Therefore the probability of occurrence of minor collision: $\frac{2^{1+zw}/2^{1+jw}}{2^{1+zw}} = \frac{1}{2^{1+jw}} = \frac{2^{-jw}}{2}$.

5.3 The Effect of Probability of Minor Collisions on Fixed Sequence Window

The Probability of major collisions $= 2^{-jw+1}, j \in [1, z-2]$. Mathematically we find the probability of occurrence of minor collision: $2^{\frac{-jw}{2}}, j \in [1, z]$.

In the existing system according to major collisions the condition of major collision depends on the least significant $jw+1$ bits of g the probability of occurrence of this collision is around 2^{-jw+1} for both values of h_{jw} which are expected to be equally likely. The condition of minor collision depends on the value of the least significant jw bits of length. The probability of the occurrence of these collisions is around $\frac{2^{-jw}}{2}$ for the both values of h_{jw} which are expected to be equally likely.

In the existing system, experiments are conducted iteratively for n collisions and find out the length of bits according to n. When we taken the window length $=$ 4, excluding the significant bits of w and the values of k^j the collisions average is 63%. The maximum number of collisions t are varied from higher towards the middle iterations, then first and last iterations from 12 for 40 bit integers and 23 for 50 bit integers. According to [8], it is not mathematically proved when the probability was reduced or increased according to the major collisions depends upon the value of the least significant jw bits of length. The probability of minor collisions is reduced up to 50% compared with that of major collisions. It is proved that when $z \in [1, n]$ the minor collisions are reduced 50% of major collisions.

6 Conclusion

We have presented a background on elliptic curve cryptosystems (ECCs) along with the different methods used to compute the scalar multiplication (ECSM), which is the core operation of ECCs, and the various costs associated with them. According to the existing system it is not proved the probability of minor collisions increased or reduced of those major collisions depends on the value of the least significant bits of jw length. We proved that mathematically the minor collisions are reduced up to 50% of major collisions.

References

[1] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," *Advances in Cryptology - Crypto' 99*, LNCS 1666, pp. 398-412, Springer-Verlag, 1999.

[2] Z. Chen, L. Xu, and C. Wu, "Construction of large families of pseudorandom subsets of the set $\{1, 2, \cdots, N\}$ using elliptic curves," *International Journal of Network Security*, vol. 11, no. 3, pp. 149-154, 2010.

[3] M. Ciet, *Aspects of Fast and Secure Arithmetics for Elliptic Curve Cryp tography*, Ph.D. thesis, Universitée Catholique de Louvain, 2003.

[4] M. Ciet, and M. Joye, "(Virtually) free randomization techniques for elliptic curve cryptography," *Information and Communications Security - ICICS' 03*, LNCS 2836, pp. 348-359, Springer-Verlag, 2003.

[5] M. Ciet, J. J. Quisquater, and F. Sica, "Preventing differential analysis in GLV elliptic curve scalar multiplication," *Cryptographic Hardware and Embedded Systems - CHES' 02*, LNCS 2523, pp. 540-550, Springer- Verlag, 2003.

[6] C. Clavier, and M. Joye, "Universal exponentiation algorithm a first step towards provable SPA-resistance," *Cryptographic Hardware and Embedded Systems - CHES' 01*, LNCS 2162, pp. 300-308, Springer-Verlag, 2001.

[7] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," *Cryptographic Hardware and Embedded Systems - CHES' 99*, LNCS 1717, pp. 292-302, Springer-Verlag, 1999.

[8] N. M. Ebied, *Key Randomization Counter Measures To Power Analysis Attacks on Elliptic Curve Cryptosystems*, Ph.D. thesis, University of Waterloo, Ontario, Canada, 2007

[9] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no.4, pp. 469- 472, 1985.

[10] V. C. Hamacher, Z. G. Vranesic, and S. G. Zaky, *Computer Organization*, McGraw-Hill, fifth ed., 2002.

[11] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography,* Springer-Verlag, 2004.

[12] C. Heuberger, and H. Prodinger, *Personal communication*, Aug. 2003.

[13] M. Joye, and K. Villegas, "A protected division algorithm," *Smart Card Research and Advanced Applications - CARDIS' 02*, pp. 59-68, Usenix Association, 2002.

[14] M. Joy, *Defenes Against Side Channel Analysis*, Advances in Elliptic Curve Cryptography, Chap5, Cambridge University Press, 2005.

[15] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Advances in Cryptology - CRYPTO' 99*, LNCS 1666, pp. 388-397, Springer-Verlag, 1999.

[16] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smart cards," *USENIX Workshop on Smart- card Technology*, pp. 151-161. May 1999.

[17] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541-552, May 2002.

[18] A. Miyaji, T. Ono, and H. Cohen, "Efficient elliptic curve exponentiation," *Information and Communication Security, First International Conference - CICS' 97*, LNCS 1334, pp. 282-290, Springer-Verlag, 1997.

[19] B. Moller, "Securing elliptic curve point multiplication against side channel attacks," *International Security Conference - ISC' 01*, LNCS 2200, pp. 324-334, Springer-Verlag, 2001.

[20] P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Mathematics of Computation*, no. 48, pp. 243-264, 1987.

[21] K. Okeya, and T. Takagi, "The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks," *Topics in Cryptology - CT-RSA' 03*, LNCS 2612, pp. 328- 343, Springer-Verlag, 2003.

[22] G. W. Reitwiesner, "Binary Arithmetic," *Advances in Computers*, vol. 1, pp. 231-308, 1960,

[23] H. Sahu and B. K. Sharma, "An MSS based on the elliptic curve cryptosystem," *International Journal of Network Security*, vol. 11, no. 2, pp. 118-120, 2010.

[24] J. A. Solinas, "Efficient arithmetic on Koblitz curves," *Designs, Codes and Cryptography*, vol. 19, pp. 195-249, 2000.

[25] N. T$\acute{h}$ eriault, "SPA resistant left-to-right integer recordings," *Selected Areas in Cryptography - SAC' 05*, LNCS 3897, pp. 345-358, Springer-Verlag, 2006.

[26] D. Yong, Y. F. Hong, W. T. Wang, Y. Y. Zhou, and X. Y. Zhao, "Speeding scalar multiplication of elliptic curve over $GF(2^m n)$," *International Journal of Network Security*, vol. 11, no. 2, pp. 70-77, 2010.

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 SciTech 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, Tirupati and M.Phil 2nd Class from Madurai Kamaraj University, Madurai. Now I am doing 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.