Power Analysis Attacks and Algorithmic Approaches to their Countermeasures for Koblitz Curve Cryptosystems

M. Anwar Hasan

Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada
ahasan@ece.uwaterloo.ca

Abstract. Because of their shorter key sizes, cryptosystems based on elliptic curves are being increasingly used in practical applications. A special class of elliptic curves, namely, Koblitz curves, offers an additional but crucial advantage of considerably reduced processing time. In this article, power analysis attacks are applied to cryptosystems that use scalar multiplication on Koblitz curves. Both the *simple* and the *differential* power analysis attacks are considered and a number of countermeasures are suggested. While the proposed countermeasures against the simple power analysis attacks rely on making the power consumption for the elliptic curve scalar multiplication independent of the secret key, those for the differential power analysis attacks depend on randomizing the secret key prior to each execution of the scalar multiplication.

1 Introduction

If cryptographic systems are not designed properly, they may leak information that is often correlated to the secret key. Attackers who can access this leaked information are able to recover the secret key and break the cryptosystem with reasonable efforts and resources. In the recent past, attacks have been proposed that use the leaked side channel information such as timing measurement, power consumption and faulty hardware (see for example [1], [2], [3], [4], [5], [6]). These attacks are more related to the implementation aspects of cryptosystems and are different from the ones that are based on statistical properties of the cryptographic algorithms (i.e., differential and linear cryptanalysis attacks [7], [8], [9]).

In [2] and [4], Kocher et al. have presented attacks based on *simple* and *differential* power analysis (referred to as SPA and DPA respectively) to recover the secret key by monitoring and analyzing the power consumption signals. Such power signals can provide useful side channel information to the attackers. In [10], Kelsey shows how little side channel information is needed by an attacker to break a cryptosystem. In [3], Messerges et al. show how the side channel information can be maximized. In order to implement a good cryptosystem, the designer needs to be aware of such threats.

In [11], a number of power analysis attacks against smartcard implementations of modular exponentiation algorithms have been described. In [12], power analysis attacks have been extended to elliptic curve (EC) cryptosystems, where both the SPA and the DPA attacks have been considered and a number of countermeasures including private key randomization and EC point blinding have been proposed. Generic methods to counteract power analysis attacks have been reported in [13].

Cryptosystems based on a special class of ECs, referred to as anomalous binary or Koblitz curves, were proposed by Koblitz in [14]. Such cryptosystems offer significant advantage in terms of reduced processing time. The latter, along with shorter key sizes, has made Koblitz curve (KC) based cryptosystems attractive for practical applications. However, the countermeasures available for random EC based cryptosystems do not appear to be the best solution to KC based cryptosystems.

In this article power analysis attacks are investigated in the context of KC based cryptosystems. The SPA attack is considered and its countermeasures at the algorithmic level are given. The proposed countermeasures rely on making the power consumption for the elliptic curve scalar multiplication independent of the secret key. Cryptosystems equipped with such countermeasures are however not secure enough against a stronger attack based on the DPA. In this article we also consider the DPA attack and describe how an attacker can maximize the differential signal used for the power correlation. To prevent DPA attacks against KC cryptosystems, we suggest a number of countermeasures which are the main results of this article. These countermeasures depend on randomizing the secret key prior to each execution of the scalar multiplication. They are suitable for hardware implementation, and compared to the countermeasures available in the open literature, their implementation appears to be less complex.

2 Preliminaries

An elliptic curve (EC) is the set of points satisfying a bivariate cubic equation over a field. For the finite field GF(2^n) of characteristic two, the standard equation for an EC is the Weierstrass equation

$$y^2 + xy = x^3 + ax^2 + b \tag{1}$$

where $a, b \in \mathrm{GF}(2^n)$ and $b \neq 0$. The points on the curve are of the form $P = (x, y)$, where x and y are elements of GF(2^n). Let E be the elliptic curve consisting of the solutions (x, y) to equation (1), along with a special point $\mathcal{O}$ called the point at *infinity*. It is well known that the set of points on E forms a commutative finite group under the following addition operation. (More on it can be found in [15], [16], [17], [18], and [19].)

Elliptic Curve Addition: Let $P = (x, y) \neq \mathcal{O}$ be a point on E. The inverse of P is defined as $-P = (x, x + y)$. The point $\mathcal{O}$ is the group identity, i.e., $P \uplus \mathcal{O} = \mathcal{O} \uplus P = P$, were $\uplus$ denotes the elliptic curve group operation (i.e., addition). If $P_0 = (x_0, y_0) \neq \mathcal{O}$ and $P_1 = (x_1, y_1) \neq \mathcal{O}$ are two points on E and $P_0 \neq -P_1$, then the result of the addition $P_0 \uplus P_1 = P_2 = (x_2, y_2)$ is given as follows.

$$x_2 = \begin{cases} \left(\frac{y_0+y_1}{x_0+x_1}\right)^2 + \frac{y_0+y_1}{x_0+x_1} + x_0 + x_1 + a, & P_0 \neq P_1, \\ x_0^2 + \frac{b}{x_0^2}, & P_0 = P_1, \end{cases} \tag{2}$$

$$y_2 = \begin{cases} \left(\frac{y_0+y_1}{x_0+x_1}\right)(x_0 + x_2) + x_2 + y_0, & P_0 \neq P_1, \\ x_0^2 + \left(x_0 + \frac{y_0}{x_0}\right) x_2 + x_2, & P_0 = P_1. \end{cases} \tag{3}$$

The above formulas for the addition rule require a number of arithmetic operations, namely, addition, squaring, multiplication and inversion over GF(2^n). (See, e.g., [20] and [21], for efficient algorithms for finite field arithmetic.) The computational complexities of addition and squaring are much lower than those of multiplication and inversion. To simplify the complexity comparison, our forthcoming discussion in this article ignores the costs of addition and squaring operations. Also, note that the formulas in (2) and (3) for point doubling (i.e., $P_0 = P_1$) and adding (i.e., $P_0 \neq P_1$) are different. The doubling requires one inversion, two *general* multiplications and one constant multiplication, whereas the adding operation costs one inversion and two general multiplications. Since, a field inverse is several times slower than a constant multiplication, we assume that the costs of elliptic curve point doubling and adding are roughly equal. If the points on E are represented using projective coordinates, one can however expect to see a considerable difference in these two costs and needs to treat them accordingly.

Elliptic Curve Scalar Multiplication: Elliptic curve scalar multiplication is the fundamental operation in cryptographic systems based on ECs. If k is a positive integer and P is a point on E, then the scalar multiplication kP is the result of adding k copies of P, i.e.,

$$kP = \underbrace{P \uplus P \uplus \cdots \uplus P}_{k \text{ copies}},$$

and $-kP = k(-P)$. Let $(k_{l-1},\ k_{l-2},\ \cdots,\ k_1,\ k_0)_r$ be a radix r representation of k, where k_{l-1} is the most significant symbol (digit) and each k_i, for $0 \leq i < l-1$, belongs to the symbol set s used for representing k. Thus

$$\begin{aligned} kP &= \left(\sum_{i=0}^{l-1} k_i r^i\right) P = (k_{l-1} r^{l-1} P) \uplus \cdots \uplus (k_1 r P) \uplus (k_0 P) \\ &= r\left(r\left(\cdots r\left(r(k_{l-1}P) \uplus k_{l-2}P\right) \uplus \cdots\right) k_1 P\right) \uplus k_0 P. \end{aligned}$$

Then one may use the following *multiply-radix-and-add* algorithm (also known as double-and-add algorithm for $r = 2$) to compute kP in l iterations.

Algorithm 1. Scalar multiplication by multiply radix and add

```
Input: $k$ and $P$
Output: $Q = kP$
    $Q := \mathcal{O}$
    for ($j = l - 1$; $j \geq 0$; $j - -$) {
        $Q := rQ$
        if ($k_j = 1$)
            $Q := Q \uplus P$
    }
```

Remark 1. For practical purposes one can assume that k is represented with respect to the conventional binary number system (i.e., $r = 2$ and $k_j \in \{0,\ 1\}$) and that $l = n$ where n is the dimension of the underlying extension field. Then the above algorithm would require approximately $3n/2$ elliptic operations on average.

Note that the conventional binary system is non-redundant and k has only one representation. However, using a different number system which has redundancy in it, the integer k can be represented in more than one way. By choosing a representation of k that has fewer non-zeros, one can reduce the number of EC additions and hence speed-up the scalar multiplication. More on this can be found in [22] and the references therein.

Remark 2. If k is represented in the binary NAF (non-adjacent form [22]), where $r = 2$, $k_j \in \{-1, 0, 1\}$ and $k_j k_{j+1} = 0$, $0 \leq j \leq n$, then the average number of elliptic operations in Algorithm 1 is $\approx 4n/3$.

Koblitz Curves: In (1), if we set $b = 1$ and restrict a to be in $\{0, 1\}$, we have

$$y^2 + xy = x^3 + ax^2 + 1, \tag{4}$$

which gives a special class of ECs, referred to as Koblitz curves (KC). Let us denote the KC as E_a. (In the rest of this article, if a curve E as defined in conjunction with (1) is not of Koblitz type, then it is referred to as a random curve.)

In (4), since $a \in \mathrm{GF}(2)$, if (x, y) is a point on E_a, (x^2, y^2) is also a point on E_a. Using the addition rule given in the previous section, one can also verify that if $(x, y) \in E_a$, then the three points, viz., (x, y), (x^2, y^2) and (x^4, y^4) satisfy the following:

$$(x^4, y^4) \uplus 2(x, y) = (-1)^{1-a}(x^2, y^2). \tag{5}$$

Using (5), one can then obtain

$$\tau(x, y) = (x^2, y^2), \tag{6}$$

where, τ is a complex number which satisfies

$$\tau^2 - (-1)^{1-a}\tau + 2 = 0. \tag{7}$$

Equation (6) is referred to as the Frobenius map over GF(2). One important implication of (6) is that the multiplication of a point on E_a by the complex number τ can simply be realized with the squaring of the x and y coordinates of the point. As a result, if the scalar k is represented with radix τ and $k_i \in \{0, 1\}$, the above multiply-radix-and-add algorithm still can be used with $r = \tau$. The operation $Q := rQ$ would however then correspond to two squaring operations over GF(2^n). In a *normal* basis representation, squaring is as simple as a cyclic shift of the bits of the operand. Efficient squaring algorithms using the more widely used *polynomial* basis can be found in [23] and [20].

3 SPA Attack and Its Countermeasures

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 cryptosystems based on elliptic curves. In the elliptic curve version of the Diffie-Hellman key exchange, the scalar k is the *private* key which is an binary integer of about n bits long. The security of many public-key cryptosystems depends on the secrecy of the private key. In many applications, the key is stored inside the device. In the recent past, attacks have been reported in the open literature to recover the key by analyzing the power consumption of cryptosystems. Following [12], here we first briefly describe the simple power analysis (SPA) attack and a countermeasure against it (denoted as *simple countermeasure*). Although the countermeasure, which is a close-variant of its counterpart in modular exponentiation, is easy to implement, below we show that its straight-forward implementation gives away the computational advantage one would expect from the use of Koblitz curves. We then discuss two simple modifications for possible improvements.

3.1 SPA Attack

In general, power analysis attacks rely on the difference between power consumptions of the cryptosystem when the value of a specific partitioning function is above and below a suitable threshold. For example, when a cryptosystem is performing a simple operation (such as the Frobenius map), the power consumption may be related to the Hamming weight of the operand. Large differences in power consumptions may be identified visually or by simple analysis.

If the scalar multiplication is performed using the *multiply-radix-and-add* algorithm, a simple power analysis can be applied. In Algorithm 1, the operation $Q := rQ$ is performed in each iteration irrespective of the value of k_j. However, the step with $Q := Q \uplus P$ is processed if $k_j = 1$, which requires a number of time and power consuming operations, such as, GF(2^n) multiplication and inversion as shown in (2) and (3). This enables an attacker to easily analyze the power consumption signals, especially to detect the difference in power consumption (and time) and to eventually recover k_j. An attacker may need as low as one iteration of the multiply-radix-and-add algorithm to obtain k_j.

3.2 Coron's Simple Countermeasure

A straightforward countermeasure for the SPA attack is to make the execution of the elliptic curve addition independent of the value of k_j. This can be achieved by performing the elliptic addition in each iteration irrespective of the value of k_j and use the sum in the subsequent steps as needed. This is shown in the following algorithm. The latter, along with r replaced by 2, yields the SPA resistant scalar multiplication for random curves proposed by Coron [12].

Algorithm 2. SPA resistant scalar multiplication
Input: k and P
Output: $Q = kP$

```
for $(j = l-1; j >= 0; j--)\{$
    $Q[0] := rQ[0]$
    $Q[1] := Q[0] \uplus P$
    $Q[0] := Q[k_j]$
$\}$
Q:=Q[0]
```

Assuming that the difference in power consumption to access $Q[0]$ and $Q[1]$ is negligible, the power consumption for executing $Q[0] := Q[k_j]$ (and hence the above algorithm) does not depend of the value of k_j. As a result, the simple power analysis attack would not be effective to recover k.

As mentioned earlier, for the binary representation of k, the latter can be n bits long implying that the above algorithm would require $2n$ elliptic operations (doubling and adding). On the other hand, to take advantage of the simple Frobenius mapping associated with the Koblitz curve, k is usually represented with radix $r = \tau$. For such τ-adic representation, if we limit k_j, for $0 \le j \le l-1$ to be 0 and 1 only, then the value of l in Algorithm 2 can be $\approx 2n$ [24]. Thus, the algorithm would require about $2n$ elliptic operations (only addition, no doubling) and does not appear to provide computational advantages of using Koblitz curves over random curves. The following discussion however attempts to alleviate this problem.

3.3 Reduced Complexity Countermeasure

Since the solutions (x, y) to equation (4) are over GF(2^n), we have $x^{2^n} \equiv x$. Consequently,

$$\begin{aligned} \tau^n(x,y) &= (x^{2^n}, y^{2^n}) \equiv (x,y) \\ (\tau^n - 1)(x,y) &\equiv \mathcal{O}. \end{aligned} \tag{8}$$

Thus, for the scalar multiplication $Q = kP$, instead of using k, one can use $k \pmod{\tau^n - 1}$ and an n-tuple can be used to represent $k \pmod{\tau^n - 1}$ in radix τ. Let $\kappa = (\kappa_{n-1}, \kappa_{n-2}, \cdots, \kappa_0)_\tau$ denote the τ-adic representation of k (mod $\tau^n - 1$), where $\kappa_i \in s$. The latter corresponds to the set of symbols used for representing the reduced k. Efficient algorithms exist to reduce k modulo $\tau^n - 1$

(see for example [24], [25]). As it will be shown later, in certain situations, it appears to be advantageous to use an expanded symbol set which results in a redundant number system. Assume that $s = \{s_0, s_1, \cdots, s_{|s|-1}\}$ with $s_0 < s_1 < \cdots < s_{|s|-1}$. For the sake of simplicity, if we also assume that s is symmetric around zero (e.g., $s = \{-1,\ 0,\ 1\}$), the following algorithm for $Q = kP$ is SPA resistant.

Algorithm 2a. SPA resistant scalar multiplication with reduced τ representation
Input: k and P
Output: $Q = kP$

```
    for $(j = n-1; j >= 0; j--)\{$
        $Q[0] := \tau Q[0]$
        $Q[1] := Q[0] \uplus P;$    $Q[-1] := -Q[1]$
        $Q[2] := Q[1] \uplus P;$    $Q[-2] := -Q[2]$
        $\vdots$
        $Q[(|s|-1)/2] := Q[(|s|-3)/2] \uplus P;$    $Q[-(|s|-1)/2] := -Q[(|s|-1)/2]$
            /* $|s|$ is odd for symmetric $s$ */
        $Q[0] := Q[\kappa_j]$
    $\}$
    Q = Q[0]
```

The cost of calculating the additive inverse of an elliptic point is simply equal to the addition of two elements of GF(2^n) and it is quite small compared to an elliptic addition. As a result, the computational cost of the above algorithm is essentially $n(|s|-1)/2$ elliptic operations (additions only). In terms of storage requirements, the above algorithm uses buffers to hold $|s|$ elliptic points. These buffers are accessed in each iteration. For high speed cryptosystems, these points can be buffered in the registers of the underlying processor. For practical applications, the number of registers needed for this purpose may be too high for many of today's processors. For example, if a Koblitz curve over GF(2^{163}), recommended by various standardization committees, is used, then twelve 32-bit registers are needed to hold a single elliptic point (both x and y coordinates in uncompressed form). Assuming that there are only three symbols in the set s, the total number of 32-bit registers needed is thirty six.

Remark 3. If k is reduced mod $\tau^n - 1$ and represented in the signed binary τ-adic form [22], where $r = \tau$ and $s = \{-1, 0, 1\}$ and then the number of elliptic operations (i.e., additions) in Algorithm 2a. is n.

3.4 Countermeasure with Large Sized Symbol Set

With the increase of $|s|$, the cost of this algorithm increases linearly and the advantage of using the Frobenius map, rather than the *point doubling*, diminishes. What follows below is a way to reduce the cost of scalar multiplication for Koblitz curves using a few pre-computed elliptic curve points.

From (8), one can write

$$(\tau - 1)(\tau^{n-1} + \tau^{n-2} + \cdots + 1)(x, y) \equiv \mathcal{O}$$

implying that

$$(\tau^{n-1} + \tau^{n-2} + \cdots + 1)(x, y) \equiv \mathcal{O}. \tag{9}$$

Thus, in the context of the scalar multiplication one has

$$k \ (\mathrm{mod}\ \tau^n - 1) \equiv \sum_{i=0}^{n-1} (\kappa_i - s_0 + 1)\tau^i$$

where s_0 is the smallest integer in s. Notice that $\kappa_i - s_0 + 1$ ensures that each symbol of the reduced k representation is a non-zero positive integer. Now, we have an efficient way to compute scalar multiplication as follows.

Algorithm 2b. Efficient SPA resistant scalar multiplication with reduced τ representation
Input: k and P
Output: $Q = kP$

```
    $P[0] = \mathcal{O}$
    for $(i = 0; i \le |s| - 1; i++)\{$
        $P[i+1] := P[i] \uplus P$
    $\}$
    $Q := \mathcal{O}$
    for $(j = n - 1; j >= 0; j--)\{$
        $Q := \tau Q \uplus P[\kappa_j - s_0 + 1]$
    $\}$
```

This algorithm requires a maximum of $n + |s|$ elliptic operations (in contrast to $n(|s|-1)/2$ elliptic operations in Algorithm 2a.). More importantly, the number of registers needed in the loop of the above algorithm does not increase with the increase of the symbol set size. This may be advantageous for register constrained processors. The pre-computed points, namely, $P[i]$, for $0 \le i \le |s| - 1$, can be stored in a RAM. The latter is updated at the beginning of the algorithm and is accessed only once in each iteration.

If the activities on the address bus of the RAM can be monitored, it can be used by the attacker to reduce his efforts to recover κ_j. One way to prevent this kind of information leakage is to load the pre-computed points into random locations inside the RAM.

4 DPA Attack

SPA attacks would fail when the differences in the power signals are so small that it is infeasible to directly observe them and to apply simple power analysis. In such cases, an attacker can apply differential power analysis (DPA). The

DPA attacks are based on the same underlying principle of SPA attacks, but use statistical and digital signal processing techniques on a large number of power consumption signals to reduce noise and to strengthen the *differential* signal. The latter corresponds to the peak, if any, in the power correlation process. This signal is an indication whether or not the attacker's guess about a symbol of the n-tuple representation of the secret key is correct.

Kocher et al. first introduced the idea of DPA to attack DES [2], [4]. This DPA attack was strengthened by Messerges et al. in [3]. Coron applied the DPA attack against EC cryptosystems [12]. In this section, this attack is briefly described and possible strategies that the attacker can use to strengthen the differential signal are presented.

4.1 DPA Attack on EC Scalar Multiplication

Assume that a cryptosystem uses one of the scalar multiplication algorithms described in the previous section. Although, Algorithms 2, 2a. and 2b. are all SPA resistant, and iterations of each of these algorithms have equal amount of computational load irrespective of k_j or κ_i, the latter remains the same in all runs of the algorithm. One can take advantage of this to recover the scalar in the DPA attack as follows.

In order to apply the DPA attack, the algorithm is executed repeatedly (say, t times) with points $P_0, P_1, \cdots, P_{t-1}$ as inputs. During the execution of the algorithm, the power consumption is monitored for each iteration in which two elliptic curve points are added. In its i-th execution of the algorithm, let the power consumption signal of the j-th iteration be $S_{i,j}$, $0 \leq i \leq t-1$ and $0 \leq j \leq n-1$. Assume that the most significant $n-j'-1$ symbols, namely, $\kappa_{n-1}, \kappa_{n-2}, \cdots, \kappa_{j'+1}$ are known. In order to determine the next most significant symbol $\kappa_{j'}$, the attacker proceeds as follows.

In an attempt to analyze the power signals, a partitioning function is chosen by the attacker. This function, in its simplest form, is the same for all j and is of two value logic. The function's value depends on k, more specifically, for $j = j'$, it depends on $\kappa_{n-1}, \kappa_{n-2}, \cdots, \kappa_{j'}$. The true value, which is still unknown to the attacker, is generated within the devise which executes the scalar multiplication algorithm. Let us denote this value as $\gamma_{i,j'} \in \{0,1\}, 0 \leq i \leq t-1$. For the DPA to work for the attacker, there ought to be a difference in the power consumptions based on the two values. By guessing a value (say $\kappa'_{j'}$) for $\kappa_{j'}$, the attacker

1. comes up with his own value $\gamma'_{i,j'}$, $0 \leq i \leq t-1$, for the partitioning function,
2. splits the power signals $S_{i,j'}$, for $0 \leq i \leq t-1$, into two sets: $S_0 = \{S_{i,j'} | \gamma'_{i,j'} = 0\}$ and $S_1 = \{S_{i,j'} | \gamma'_{i,j'} = 1\}$, and finally
3. computes the following differential signal

$$\delta(j') = \frac{\sum\limits_{S_{i,j'} \in S_0} S_{i,j'}}{\sum\limits_{i=0}^{t-1} \gamma'_{i,j'}} - \frac{\sum\limits_{S_{i,j'} \in S_1} S_{i,j'}}{\sum\limits_{i=0}^{t-1} (1 - \gamma'_{i,j'})}. \tag{10}$$

Notice that

$$\Pr(\gamma'_{i,j'} = \gamma_{i,j'}) \approx \begin{cases} \frac{1}{2}, \text{ if } \kappa'_{j'} \neq \kappa_{j'} \\ 1, \text{ if } \kappa'_{j'} = \kappa_{j'} \end{cases} \tag{11}$$

and for a sufficiently large value of t,

$$\lim_{t\to\infty} \delta(j') \approx \begin{cases} 0, \text{ if } \kappa'_{j'} \neq \kappa_{j'} \\ \epsilon, \text{ if } \kappa'_{j'} = \kappa_{j'} \end{cases} \tag{12}$$

where ϵ is related to the difference of the average power consumptions with $\gamma_{i,j'}$ being 0 and 1. This non-zero value of the differential signal indicates a correct guess for $\kappa_{j'}$. For the symbol set s, to obtain a non-zero differential signal the attacker needs to perform the differential power analysis $|s|/2$ times, on average.

5 Countermeasures against DPA Attacks

In this section we describe three countermeasures to prevent the DPA that an attacker can use in an effort to learn the (secret) scalar k of the Koblitz curve scalar multiplication $Q = kP$. The underlying principle is that if k is randomly changed each time it is used in the cryptosystem, the averaging out technique used in the DPA would not converge to an identifiable differential signal and the DPA attacks are expected to fail. The main challenge however is to change k to pseudo-random values with a reasonable cost and still providing the same Q.

The countermeasures presented below can be applied separately. However, when they are used together, one can expect to attain highest level of protection against power analysis attacks.

5.1 Key Masking with Localized Operations (KMLO)

For the sake of simplicity, in (4) assume that $a = 1$ (an extension using $a = 0$ is straight-forward). Then, using (7) one can write

$$2 = \tau - \tau^2 = -\tau^3 - \tau, \tag{13}$$

which shows two different representations of '2'. This in turn allows the τ-adic symbols of k to be replaced in more than one way on a window of three or more symbols. For example, using the above two representations of '2', the window of four symbols vis., $(\kappa_{i+3}, \kappa_{i+2}, \kappa_{i+1}, \kappa_i)$ can be replaced by $(\kappa_{i+3} \pm d_{i+3}, \kappa_{i+2} \pm d_{i+1}, \kappa_{i+1} \pm d_{i+1}, \kappa_i \pm d_i)$ where

$$\begin{array}{rl} & (d_{i+3},\ d_{i+2},\ d_{i+1},\ d_i) \\ = & (0,\ \ \overline{1},\ \ 1,\ \ \overline{2}) \\ = & (\overline{1},\ \ 0,\ \ \overline{1},\ \ 2) \\ = & (\overline{1},\ \ 1,\ \ \overline{2},\ \ 0) \end{array} \tag{14}$$

and $\overline{z} = -z$. If d_i's are allowed to take values outside the range [-2, 2], more combinations can be obtained. These combinations can be used to modify the symbols of the window such that the resultant symbols belong to an expanded set s. Also, the windows can be overlapped, their starting positions can be changed and their sizes can be varied.

To implement this key masking scheme in hardware, one can use an n-stage shift register where each stage can hold one symbol of s. The key k reduced modulo $\tau^n - 1$ is initially loaded into the register. A masking unit will take a w-tuple vector ($w \geq 3$) consisting of any w adjacent symbols from the register and adds it to another vector derived from (13). (For $w = 4$, a set of possible vectors are given in (14).) The resultant vector then replaces the original w-tuple vector in the register. This process is repeated by shifting the contents of the register, possibly to mask all the symbols stored in the register. During this masking process, if a resultant symbol lies out side the set s, one can repeatedly apply (13) to restrict the symbol within s. Additionally, since $(\tau^{n-1}+\tau^{n-2}+\cdots+1)P \equiv 0P$,

$$Q = \left(\sum_{i=0}^{n-1} \kappa_i \tau^i \right) P \equiv \left(\sum_{i=0}^{n-1} \hat{\kappa}_i \tau^i \right) P \tag{15}$$

where $\hat{\kappa}_i = \kappa_i \pm c$, for $0 \leq i \leq n-1$, and c is an integer. Hence, a bias can be applied to each symbol of the key without any long addition (and hence without any carry propagation).

5.2 Random Rotation of Key (RRK)

Let

$$P' = \tau^r P, \tag{16}$$

where r is a random integer such that $0 \leq r \leq n-1$. Using (8), the elliptic curve scalar multiplication can be written as follows:

$$\begin{aligned}
Q &= kP \\
&= (\kappa_{n-1}\tau^{n-1} + \kappa_{n-2}\tau^{n-2} + \cdots + \kappa_0)P \\
&= \tau^{r-n} \left(\kappa_{r-1}\tau^{n-1} + \kappa_{r-2}\tau^{n-2} + \cdots + \kappa_r \right) P \\
&= \left(\kappa_{r-1}\tau^{n-1} + \kappa_{r-2}\tau^{n-2} + \cdots + \kappa_r \right) P' = \left(\sum_{i=0}^{n-1} \kappa_{(r-1-i) \bmod n}\ \tau^i \right) P' \\
&= \tau\,(\ \tau\,(\ \cdots \tau\,(\ \tau(\kappa_{r-1}P') \uplus \kappa_{r-2}P') \uplus \cdots) \uplus \kappa_{r+1}P') \uplus \kappa_r P'.
\end{aligned}$$

This leads to the following algorithm, where the operation "$P'[i] := s_i P'$ " can be replaced by "$P'[i] := (s_i - s_0 + 1)P'$ " if an elliptic addition of $\mathcal{O}$ has to be avoided.

Algorithm 3. Scalar Multiplication with Key Rotation
Input: k and P
Output: $Q = kP$

```
    Pick up a random number $r \in \{0,\ 1,\ \cdots,\ n-1\}$
    Compute $P' = \tau^r P$
```

```
$Q := \mathcal{O}$; $j = r$; $P'[i] := s_i P'$,   $0 \le i \le |s| - 1$
for ($i = 0$; $i \le n - 1$; $i$++) {
    $j := j - 1 \pmod{n}$
    $Q := \tau Q$
    $Q := Q \uplus P'[\kappa_j - s_0]$
}
```

Before starting the iterations, Algorithm 3 computes P'. From (16), we have

$$P' = (x^{2^r}, y^{2^r}) \tag{17}$$

where x and y are the coordinates of P. Let the *normal* basis representations of x and y be

$$\begin{aligned} x &= (x_{n-1},\ x_{n-2},\ \cdots,\ x_0) \quad \text{and} \\ y &= (y_{n-1},\ y_{n-2},\ \cdots,\ y_0), \end{aligned}$$

respectively. Then, one can write

$$\begin{aligned} x^{2^r} &= (x_{n-r-1},\ x_{n-r-2},\ \cdots,\ x_0, x_{n-1}, \cdots,\ x_{n-r}) \quad \text{and} \\ y^{2^r} &= (y_{n-r-1},\ y_{n-r-2},\ \cdots,\ y_0, y_{n-1}, \cdots,\ y_{n-r}) \end{aligned}$$

which correspond to r-fold left cyclic shift of the representations of x and y, respectively. Thus, using a normal basis representation, one can easily compute $P' = \tau^r P$ with minimal risk of revealing the value of r against power attacks.

On the other hand, if x and y are represented with respect to a *polynomial* or other basis where the τ^r mapping is accomplished in an iterative way, measures must be taken so that the number of iterations does not reveal r.

5.3 Random Insertion of Redundant Symbols (RIRS)

The basis of this method for key randomization is that before each scalar multiplication a number of redundant symbols are inserted at random locations in the secret key sequence. In order to correctly generate the shared secret, these redundant symbols however must collectively nullify their own effects.

Before a particular scalar multiplication, assume that a total of n' redundant symbols denoted as f_i, for $0 \le i \le n' - 1$, are inserted into the original key sequence $\{\kappa_i\}$, for $0 \le i \le n - 1$. Thus the resultant sequence, denoted as $b = \{b_i\}$, for $0 \le i \le N - 1$, has $N = n + n'$ symbols. When an SPA resistant algorithm (like the ones in Section 3) is applied to do a scalar multiplication, the redundant symbols are paired, i.e., (f_0, f_1), (f_2, f_3), $\cdots$, $(f_{n'-2}, f_{n'-1})$ where n' is even. For the sake of simple implementation, redundant pairs are picked up in sequence, (i.e., first (f_0, f_1), then (f_2, f_3) and so on) and inserted at random adjacent locations starting from the most significant symbol of b. For example, with $n' \ge 4$, if the pair (f_{2l}, f_{2l+1}), for $0 \le l < n'/2 - 1$, is inserted at locations i and $i - 1$, then (f_{2l+2}, f_{2l+3}) is inserted at j and $j - 1$, for some i and j such that $N - 1 \ge i > j > 1$.

In order to implement this scheme, an N bit random number g is generated which has $\frac{n'}{2}$ non-adjacent one's. The locations of these 1's are aligned with the redundant symbols f_{2l+1}, for $0 \leq l \leq n'/2-1$, each of which corresponds to the second element of a pair of redundant symbols as shown below.

$$\begin{array}{rcccccccc} \text{Location}: & N-1 & \cdots & N-r+1 & N-r & N-r-1 & N-r-2 & \cdots \\ g: & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots \\ b: & \kappa_{n-1} & \cdots & \kappa_{n-r+1} & f_0 & f_1 & \kappa_{n-r} & \cdots \end{array} \quad (18)$$

In (18), the first pair (f_0, f_1) has been inserted at random location $N-r$. Assume that all b_i's belong an *expanded* symbol set $u = \{u_0, u_1, \cdots, u_{|u|-1}\}$. To perform scalar multiplication on the Koblitz curve, we can then state the following algorithm, where b'_i is an integer in the range [0,$|u|$-1] and is obtained as $b'_i = \imath$, for $0 \leq \imath \leq |u|-1$, given that $b_i = u_\imath$.

Algorithm 4. Scalar Multiplication with Random Insertions
Input: k and P
Output: $Q = kP$

```
Compute $P[i] = u_i P$,    $0 \leq i \leq |u|-1$
Generate $g$ and form $b$
$Q := \mathcal{O}$; $R[0] := \mathcal{O}$; $R[1] := \mathcal{O}$
for $(i = N-1;\ i \geq 0;\ i--)$ {
    $R[0] := \tau R$,   $R[1] := \tau^{-1} R$
    $Q := R[g_i] \uplus P[b'_i]$
}
```

In the above algorithm, let $Q^{(j)}$ denote the point Q at iteration $i = j$. In order to determine the value of the redundant symbols, without loss of generality we refer to (18). To obtain the correct Q at the end of iteration $i = N-r+2$, the algorithm should yield

$$Q^{(N-r-2)} = \kappa_{n-1}\tau^{r-1} + \kappa_{n-2}\tau^{r-2} + \cdots + \kappa_{n-r+1}\tau + \kappa_{n-r}. \quad (19)$$

In this regard, note that

$$Q^{(N-r)} = \kappa_{n-1}\tau^{r-1} + \kappa_{n-2}\tau^{r-2} + \cdots + \kappa_{n-r+1}\tau + f_0.$$

With $i = N-r-1$ and $i = N-r-2$ we have $g_{N-r-1} = 1$ and $g_{N-r-2} = 0$, respectively. In Algorithm 4, these two values of g_i's correspond to τ^{-1} and τ mappings, respectively, and yield the following:

$$Q^{(N-r-1)} = \tau^{-1}(\kappa_{n-1}\tau^{r-1} + \kappa_{n-2}\tau^{r-2} + \cdots + \kappa_{n-r+1}\tau + f_0) + f_1,$$
$$Q^{(N-r-2)} = \kappa_{n-1}\tau^{r-1} + \kappa_{n-2}\tau^{r-2} + \cdots + \kappa_{n-r+1}\tau + f_0 + f_1\tau + \kappa_{n-r}. \quad (20)$$

Comparing (19) and (20), one can see that if f_1 is chosen to nullify the effect of f_0, then these two redundant symbols should have the following relationship

$$f_0 + \tau f_1 = 0. \quad (21)$$

On the other hand, if the pair (f_2, f_3) is to cancel (f_0, f_1), then the following should satisfy

$$f_2 + \tau f_3 + \tau^{m-1}(f_0 + \tau f_1) = 0 \tag{22}$$

assuming that f_2 is m positions away from f_1. The value of m can be varied. However for the sake of simpler implementation it can be fixed to a value which could be as small as unity.

5.4 Comments

- The three methods presented above use τ-adic representation of the key k. The first and the third methods can be extended to other bases of presentation of k.
- Each of the three proposed methods uses a look-up table which hold the points u_iP, for $0 \leq i \leq |u| - 1$, corresponding to the expanded symbol set u which contains all the symbols of the randomized key. The table contents are to be updated each time a new P (or P' in the random rotation of the key) is used. If the symbol set is symmetric then the look-up table may contain only $\{u_iP : u_i \geq 0, 0 \leq i \leq |u| - 1\}$. This is possible because the negative multiples of P can be obtained from the positive multiples using $-(x, y) = (x, x + y)$. Although, this technique reduces the table size by half, it introduces an extra step for the negative digits, which may reveal the signs of the digits unless proper measures are taken to protect them.

6 Comparison and Concluding Remarks

In the randomization technique of [12], a multiple of the total number of curve points $\mathcal{E}$ is added to k. Since $\mathcal{E}P = \mathcal{O}$,

$$Q = kP = (k + e\mathcal{E})P \tag{23}$$

where e is an integer. The realization of this key randomization scheme requires a large integer multiplier ($\mathcal{E}$ is about n bits long). If this multiplier is not already part of the system into which the power analysis resistant elliptic curve scalar multiplication is to be embedded, it will result in a considerable increase in the silicon area. On the other hand, the key masking/randomization scheme presented in Section 5 does not need such a multiplier.

Recently, Chari et al. have proposed generic methods [13] to countermeasure differential power analysis attacks. For instance, one can randomly pick up a pair of numbers k' and k'' such that $k = k' + k''$, and computes $k'P \uplus k''P$. A straightforward implementation of this scheme would require longer computation time since two scalar multiplications are involved. In order to reduce the computation time, if certain speed-up techniques (e.g., Shamir's trick) are used, then one would require relatively more complex implementation.

Although, the method proposed in [11] focuses on key randomization in modular exponentiation, one can attempt to extend it to elliptic curve scalar multiplication. It starts the computation of Q at a random symbol of k, but always terminates the computation at the most significant symbol giving the adversary an opportunity to work backwards. For such an attack on Koblitz curve based cryptosystems, the adversary needs to compute the inverse of the τ mapping, which unlike the square root operation in modular exponentiation, can be quite simple. On the other hand, Algorithm 3 of this article also starts at random position (i.e., symbol), but unlike [11] it terminates adjacent to the random starting position. As a result, it is less vulnerable to the backward power analysis attack.

When applied to Koblitz curve based cryptosystems, the proposed countermeasures are expected to be less complex than the similar ones already exist. Nevertheless, their overall impacts on the cryptosystems need to be carefully investigated and possible trade-offs are to be identified for implementation in real systems. More importantly, these countermeasures are to be investigated against more advanced attacks.

Acknowledgment

This work was mostly done during the author's sabbatical leave with the Motorola Laboratories, Schaumburg, IL. The author thanks Larry Puhl for his encouragement to pursue this work, and acknowledges the stimulating discussions he had on this topic with Ezzy Dabbish, Tom Messerges and Brian King.

References

1. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Systems," in *Advances in Cryptology- CRYPTO '96, Lecture Notes in Computer Science*, pp. 104–113, Springer-Verlag, 1996.
2. P. Kocher, J. Jaffe, and B. Jun, "Introduction to Differential Power Analysis and Related Attacks." `http://www.cryptography.com/dpa/technical`, 1998.
3. T. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigation of Power Analysis Attacks on Smartcards," in *Proceedings of USENIX Workshop on Electronic Commerce*, pp. 151–161, 1999.
4. P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology- CRYPTO '99, Lecture Notes in Computer Science*, pp. 388–397, Springer-Verlag, 1999.
5. D. Boneh, R. A. Demillo, and R. J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," in *Advances in Cryptology- CRYPTO '97, Lecture Notes in Computer Science*, pp. 37–51, Springer-Verlag, 1997.
6. E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," in *Advances in Cryptology- CRYPTO '97, Lecture Notes in Computer Science*, pp. 513–525, Springer-Verlag, 1997.
7. E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.
8. M. Matsui, "Linear Cryptanalysis Method for DES Cipher," in *Advances in Cryptology- EUROCRYPT '93, Lecture Notes in Computer Science*, pp. 386–397, Springer-Verlag, 1994.

9. E. Biham and A. Shamir, “Differential Cryptanalysis of the Full 16-round DES,” in *Advances in Cryptology- CRYPTO ’92, Lecture Notes in Computer Science*, pp. 487–496, Springer-Verlag, 1993.
10. J. Kelsey, “Side Channel Cryptanalysis of Product Ciphers,” in *ESORICS, Lecture Notes in Computer Science*, pp. 487–496, Springer-Verlag, 1998.
11. T. Messerges, E. A. Dabbish, and R. H. Sloan, “Power Analysis Attacks on Modular Exponentiation in Smartcards,” in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, pp. 144–157, LNCS, Springer-Verlag, 1999.
12. J.-S. Coron, “Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems ,” in *Workshop on Cryptographic Hardware and Embedded Systems*, pp. 292–302, LNCS, Springer-Verlag, 1999.
13. S. Chari, J. R. R. C. S. Jutla, and P. Rohatgi, “Towards Sound Approaches to Counteract Power-Analysis Attacks,” in *Advances in Cryptology- CRYPTO ’99*, pp. 398–412, 1999.
14. N. Koblitz, “CM-Curves with Good Cryptographic Properties,” in *Advances in Cryptology- CRYPTO ’91, Lecture Notes in Computer Science*, pp. 279–287, Springer-Verlag, 1992.
15. V. S. Miller, “Use of Elliptic Curves in Cryptography,” in *Advances in Cryptology- CRYPTO ’85*, pp. 417–426, Springer, 1986.
16. N. Koblitz, “Elliptic Curve Cryptosystems,” *Math. Comp.*, vol. 48, pp. 203–209, 1993.
17. A. J. Menezes, *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.
18. J. H. Silverman, *The Arithmetic of Elliptic Curves*, vol. 106. New York: Springer-Verlag, 1986.
19. I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography.* Cambridge Univ Press, 1999.
20. R. Schroeppel, S. O’Malley, H. Orman, and O. Spatscheck, “A Fast Software Implementation for Arithmetic Operations in GF(2^n),” in *Advances in Cryptology- CRYPTO ’95, Lecture Notes in Computer Science*, pp. 43–56, Springer, 1995.
21. H. Wu, M. A. Hasan, and I. F. Blake, “Highly Regular Architectures for Finite Field Computation Using Redundant Basis,” in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, pp. 269–279, LNCS, Springer-Verlag, 1999.
22. D. Gordon, “A Survey of Fast Exponentiation Methods,” *Journal of Algorithms*, vol. 27, pp. 129–146, 1998.
23. H. Wu, “Low Complexity Bit-Parallel Finite Field Arithmetic Using Polynomial Basis,” in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, pp. 280–291, LNCS, Springer-Verlag, 1999.
24. J. Solinas, “An Improved Algorithm for Arithmetic on a Family of Elliptic Curves,” in *Advances in Cryptology- CRYPTO ’97, Lecture Notes in Computer Science*, pp. 357–371, Springer-Verlag, 1997.
25. T. Kobayashi, H. Morita, K. Kobayashi, and F. Hoshino, “Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt to Higher Characteristic,” in *Advances in Cryptology- EUROCRYPT ’99, Lecture Notes in Computer Science*, pp. 176–189, Springer-Verlag, 1999.