Power Analysis Attacks And Algorithmic Approaches To Their Countermeasures For Koblitz Curve Crypto-systems *

M. A. Hasan

Author affiliation:
Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada
Email: ahasan@ece.uwaterloo.ca
Ph: +1 519 888 4567 ext 2868; Fax: +1 519 746 3077

Abstract

Because of their shorter key sizes, crypto-systems 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 crypto-systems 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. These countermeasures are computationally efficient and suitable for hardware implementation.

Keywords Cryptography, elliptic curve scalar multiplication, finite (or Galois) fields, Koblitz curves, number system, power analysis attacks.

*Appeared in *IEEE Trans. Computers,* pp. 1071-1083, October 2001. A preliminary version of this article was presented at the Workshop on Cryptographic Hardware and Embedded Systems, Worcester, MA, August 2000 [10].

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 crypto-system 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 [15], [16], [20], [17], [5], [3]). These attacks are more related to the implementation aspects of crypto-systems and are different from the ones that are based on statistical properties of the cryptographic algorithms (i.e., differential and linear cryptanalysis attacks [1], [18], [2]).

In [16] and [17], 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. As shown in Figure 1, the secret key is stored in a crypto-system (e.g., a smart-card), which relies on external power supply and the attacker has the ability to monitor the power consumption signals. Such signals can provide the attacker with useful side channel information that may eventually

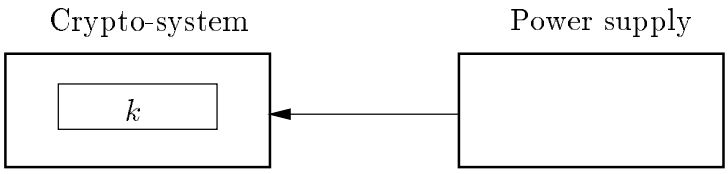

Figure 1: Block diagram showing a crypto-system which has its secret key stored inside and relies on an external power source.

reveal the key. In [11], Kelsey shows how little side channel information is needed by an attacker to break the crypto-system. In [20], Messerges et al. show how the side channel information can be maximized. In order to implement a good crypto-system, the designer needs to be aware of such threats.

In [21], a number of power analysis attacks against smart-card implementations of modular exponentiation algorithms have been described. In [8], power analysis attacks have been extended to elliptic curve (EC) crypto-systems, 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. A number of methods to counteract power analysis attacks have also been reported in [7].

In the recent past, for cryptographic applications two classes of elliptic curves, namely random and Koblitz (a.k.a. anomalous binary) curves, have received considerable attention and specific parameters for these curves have been proposed by a number of organizations, such as NIST (National Institute of Standards and Technology) [26] and SECG (Standards for Efficient Cryptography Group) [6]. Crypto-systems based on Koblitz curves were proposed by Koblitz in [13]. Such crypto-systems offer significant advantage in terms of reduced processing time. The latter, along with shorter key sizes, has made Koblitz curve (KC) based crypto-systems quite attractive for practical applications. However, the existing countermeasures against power analysis attacks on random EC based crypto-systems do not appear to be the best solution to KC based crypto-systems.

In this article, power analysis attacks and countermeasures in the context of KC based crypto-systems are investigated. 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. Crypto-systems 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 crypto-systems, we suggest a number of countermeasures. These countermeasures depend on randomizing the secret key prior to each execution of the SPA resistant 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. More importantly, it is shown that their use can

attenuate the differential signal to a level which makes the DPA attack infeasible with technologies available today.

The organization of the remainder of this article is as follows. The next section briefly introduces random and Koblitz elliptic curves and their group operations. Section 3 describes the simple power analysis attack, its countermeasures and related trade-offs. The differential power analysis attack is discussed in Section 4. Also, the time and space requirements for the DPA attack, and possible strategies that the attacker can use to strengthen the differential signal are presented in this section. The DPA countermeasures along with their costs are suggested in Section 5. Section 6 presents arguments regarding the protection provided by the proposed DPA countermeasures. Section 7 gives a comparison of a number of existing similar countermeasures with the ones proposed in this article. Finally, concluding remarks are made in Section 8.

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

$$y^2 + xy = x^3 + \alpha x^2 + \beta \tag{1}$$

where $\alpha, \beta \in \mathrm{GF}(2^n)$ and $\beta \neq 0$. 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 [22], [14], [19], [24], and [4].)

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.

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

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

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., [23] and [28], 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 [4], 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 $(k_{l-1},\ k_{l-2},\ \cdots,\ k_1,\ k_0)_2$ be the binary presentation of k, i.e., $k_i \in \{0,\ 1\}$ for $0 \leq i < l-1$. Thus

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

Then one may use the well known *double-and-add* scheme to compute kP in l iterations as follows.

Algorithm 1 *Scalar multiplication by double-and-add scheme*
Input: P and k
Output: $Q = kP$

```
Q := O
for (j = l − 1; j ≥ 0; j − −) {
    Q := 2Q
    if (k_j = 1)
        Q := Q ⊎ P
}
```

Remark 1 *For practical purposes one can assume 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 [9] and the references therein.

Remark 2 *If k is represented in the binary NAF (non-adjacent form [9]), $k_j \in \{-1, 0, 1\}$ and $k_j k_{j+1} = 0$, $0 \leq j \leq n-1$, then the average number of elliptic operations in Algorithm 1 is $\approx 4n/3$.* □

Koblitz curves: In (1), if we set $\beta = 1$ and restrict α to be in $\{0, 1\}$, we have

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

which gives a special class of ECs, referred to as Koblitz curves (KC). Let us denote the KC as E_α. (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 $\alpha \in$ GF(2), if (x, y) is a point on E_α, (x^2, y^2) is also a point on E_α. Using the addition rule given in the previous section, one can also verify that if $(x, y) \in E_\alpha$, then the three points, viz., (x, y), (x^2, y^2) and (x^4, y^4) satisfy the following:

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

Using (5), one can then obtain

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

where τ is a complex number which satisfies

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

Equation (6) is referred to as the Frobenius map over GF(2). One implication of (6) is that the multiplication of a point on E_α by the complex number τ can simply be realized with the squaring of the x and y coordinates of the point. 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 [27] and [23].

A more important feature of (6) is as follows. If the scalar k is represented with radix τ, then in the computation of kP using Algorithm 1, the operation $Q := 2Q$ is replaced by $Q := \tau Q$. The latter corresponds to two squaring operations over GF(2^n) and completely eliminates much more costlier elliptic curve point doubling operations.

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 of about n bits long, is the fundamental computation performed in crypto-systems based on elliptic curves. The security of elliptic curve crypto-systems depends on the difficulty of obtaining k from P and Q. The integer k is often the private key, and in many applications, it is stored inside the crypto-system. In the recent past, attacks have been reported in the open literature to recover the key by analyzing the power consumption of crypto-systems. Following [8], here we first briefly describe the simple power analysis (SPA) attack and a countermeasure against it. Although the countermeasure, which is a close-variant of its counterpart in modular exponentiation, is easy to implement, below we show that its straightforward 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 crypto-system when the value of a specific *partitioning* function is above and below a suitable threshold. For example, when a crypto-system 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.

Consider a Koblitz curve based crypto-system which uses k as the secret key. The latter is an n bit integer, i.e., $k = \sum_{i-0}^{n-1} k_i 2^i$ with $k_i \in \{0, 1\}$. To take advantage of the simple Frobenius map in the computation of scalar multiplication, the key is converted into a τ-adic representation (for conversion algorithms, the reader may refer to [25]). Let us denote this representation as $k = \sum_{i-0}^{l-1} \kappa_i \tau^i$. If we limit κ_i to be 0 and 1 only, then $l \approx 2n$ [25].

The crypto-system takes a point $P \in E_\alpha$ as input and computes kP. For this computation, the crypto-system uses Algorithm 1 (with k_j and radix 2 replaced by κ_j and τ, respectively), and relies on an external power source as shown in Fig. 2. As shown in the figure the operation $Q := \tau Q$, which corresponds to two simple squaring operations over GF(2^n), is performed in each iteration irrespective of the value of κ_j. However, the step with $Q := Q \uplus P$ is processed if $\kappa_j = 1$, 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

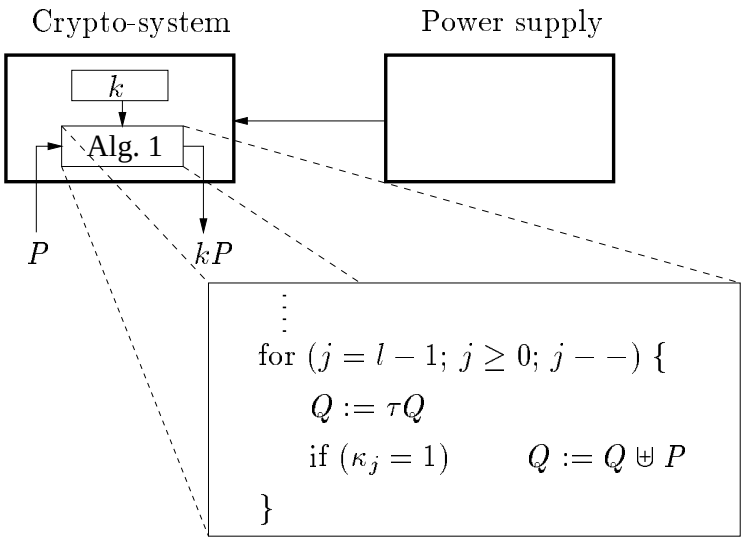

Figure 2: Scalar multiplication susceptible to SPA attack.

consumption (and time) and to eventually recover κ_j. An attacker may need as few as one iteration of the scalar multiplication algorithm to obtain κ_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 κ_j. This can be achieved by performing the elliptic addition in each iteration irrespective of the value of κ_j and use the sum in the subsequent steps as needed. This is shown in the following algorithm. In this algorithm, if radix τ is replaced by 2, we obtain the SPA resistant scalar multiplication for random elliptic curves proposed by Coron [8].

Algorithm 2 *SPA resistant scalar multiplication*
Input: P *and* $k = (\kappa_{l-1}, \cdots, \kappa_1, \kappa_0)_\tau$
Output: $Q = kP$

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

Assuming that the difference in power consumptions to access $Q[0]$ and $Q[1]$ is negligible, the power consumption for executing $Q[0] := Q[\kappa_j]$ (and hence the above algorithm) does not depend of the value of κ_j. As a result, the simple power analysis attack would not be effective to recover k. Prior to implementing this algorithm, one should confirm such equal power consumptions on the target hardware.

In the above algorithm, for the τ-adic representation of k, the loop will be executed approximately $2n$ times; hence about $2n$ elliptic operations (only additions, no doubling) are needed. On the other hand, if the original binary representation of k is used, then the Frobenius map $Q[0] := \tau Q[0]$ is to be replaced by $Q[0] := 2Q[0]$ and the loop will have only n passes; consequently, we need n additions and n doubling operations, or a total of $2n$ elliptic operations, assuming that the costs of point addition and doubling are approximately the same. Thus, Algorithm 2 does not appear to provide computational advantages to the 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 $k \pmod{\tau^n - 1}$, where $\kappa_i \in \mathcal{S}$. 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 [25], [12]). In certain situations, it appears to be advantageous to use an expanded symbol set which results in a redundant number system. Assume that $\mathcal{S} = \{s_0, s_1, \cdots, s_{|s|-1}\}$ with $s_0 < s_1 < \cdots < s_{|s|-1}$. For the sake of simplicity, if we also assume that $\mathcal{S}$ is symmetric around zero (e.g., $\mathcal{S} = \{-1,\ 0,\ 1\}$), the following algorithm for $Q = kP$ is SPA resistant.

Algorithm 2a *SPA resistant scalar multiplication with reduced τ representation*
Input: P *and* $k \equiv (\kappa_{n-1}, \kappa_{n-2}, \cdots, \kappa_0)_\tau$
Output: $Q = kP$

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

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 $n(|\mathcal{S}|-1)/2$ elliptic operations (additions only). In terms of storage requirements, the above algorithm uses buffers to hold $|\mathcal{S}|$ elliptic points. These buffers are accessed in each iteration. For high speed crypto-systems, 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 $\mathcal{S}$, 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 [9], then the number of elliptic operations (i.e., additions) in Algorithm 2a is n.*

In Table 1, we summarize the important features of scalar multiplication using the algorithms described so far. The radix used for the representation of k is denoted by r. We also use small sized symbol set suitable for resource constrained crypto-systems, such as smart-cards, which rely on external power supply. For Algorithm 1, we assume the use of the NAF for representing k, which helps to reduce the computational complexity. The NAF representation has almost no effect on the computational complexity of SPA resistant Algorithms 2 and 2a.

		Random curves		Koblitz curves	
Algorithm	SPA resistance	Representation of k	# of ellip. Ops.	Representation of k	# of ellip. Ops.
Alg. 1	No	NAF, $r = 2, \mathcal{S} = \{-1, 0, 1\}$	$4n/3$	NAF, $r = \tau$, $\mathcal{S} = \{-1, 0, 1\}$	$2n/3$
Alg. 2	Yes	$r = 2$, $\mathcal{S} = \{0, 1\}$	$2n$	$r = \tau$, $\mathcal{S} = \{0, 1\}$	$2n$
Alg. 2a	Yes	-	-	reduced mod $\tau^n - 1$, $r = \tau$, $\mathcal{S} = \{-1, 0, 1\}$	n

Table 1: Cost comparison of scalar multiplication algorithms.

3.4 Countermeasure with Large Sized Symbol Set

With the increase of $|\mathcal{S}|$, the cost of Algorithm 2a increases linearly and the advantage of using the Frobenius map over 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.

Toward this end, for $0 \leq j \leq |\mathcal{S}|$ one can compute $s_j P$ and store it in a buffer labeled as $P[s_j]$. If the operation $Q := \tau Q \uplus P[\kappa_j]$ is performed n times with j being equal to $n-1, \cdots, 1, 0$, then the final Q is the required kP. With this approach, an SPA attack can however easily recover part of the secret key k. Specifically, when $\kappa_j = 0$, we have $Q := \tau Q \uplus P[\kappa_j] = \tau Q \uplus \mathcal{O}$. Since the task of adding $\mathcal{O}$ is considerably simpler than that of any other point, the zeroes in the secret key can easily be determined by detecting the difference in respective power consumptions. To alleviate this problem, hardware techniques can be used to hide the difference. What follows below, however, is a simple scheme which achieves the same goal with almost no hardware overhead.

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

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

where s_0 is the smallest integer in $\mathcal{S}$. 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: P *and* $k \pmod{\tau^n - 1} \equiv \sum_{i=0}^{n-1} (\kappa_i - s_0 + 1)\tau^i$

Output: $Q = kP$

```
P[0] = O
for (i = 0; i <= |S| - 1; i++){
    P[i + 1] := P[i] ⊎ P
}
Q := O
for (j = n - 1; j >= 0; j--){
    Q := τQ ⊎ P[κ_j - s_0 + 1]
}
```

This algorithm requires a maximum of $n + |\mathcal{S}|$ elliptic operations (in contrast to $n(|\mathcal{S}| - 1)/2$ elliptic operations in Algorithm 2a). More importantly, the number of registers needed in the main (i.e., 2nd) 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 $0 \leq i \leq |\mathcal{S}| - 1$, can be stored in a RAM. The latter is updated at the beginning of the algorithm and is accessed only once in each iteration.

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). Kocher et al. first introduced the idea of DPA to attack DES [16], [17]. This DPA attack was strengthened by Messerges et al. in [20]. Coron applied the DPA attack against EC crypto-systems [8]. In this section, this attack is briefly described in the context of KC crypto-systems and its time and space requirements are given. Possible strategies that the attacker can use to strengthen the differential signal are presented.

4.1 DPA Attack on KC Scalar Multiplication

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 of whether or not the attacker's guess about a symbol of the n-tuple representation of the secret key is correct.

Assume that a crypto-system uses one of the SPA attack resistant scalar multiplication algorithms described in the previous section. In each of those algorithms, the representation of k (either τ-adic or reduced τ-adic) remains the same in all runs of the respective algorithm. The attacker can take advantage of this to recover the scalar in the DPA attack as discussed below.

For the sake of simplicity, assume that Algorithm 2, which is SPA resistant, is used for computing the scalar multiplication. Let $(\kappa_{l-1}, \cdots, \kappa_1, \kappa_0)_\tau$, where $\kappa_i \in \{0, 1\}$ for $0 \leq i \leq l - 1$, be the representation of the secret key k used in the algorithm. The attacker has knowledge of the algorithm and the symbol set to which the κ_i's belong. 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 (see Fig. 3). During the execution of the algorithm, the power

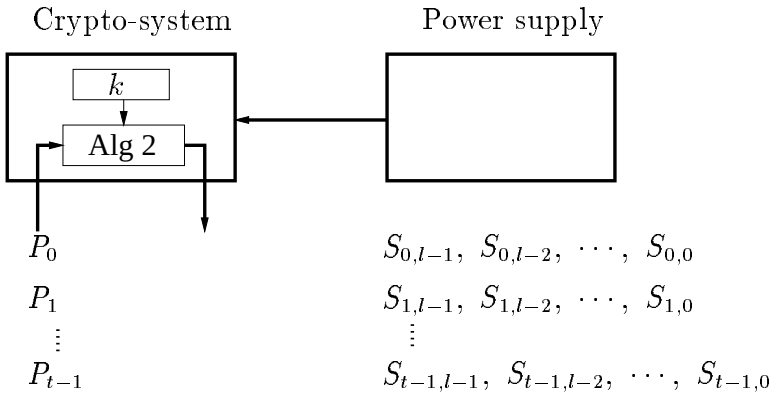

Figure 3: Input points and power consumption signals monitored in the DPA attack.

consumption is monitored for each iteration. For point P_i and the iteration where κ_j is used, let $S_{i,j}$ denote the power consumption signal monitored by the attacker. Note that Algorithm 2 uses κ_{l-1} in its first pass, then κ_{l-2} and so on. The attacker's straightforward strategy is to determine these secret bits in the same order, i.e., first obtain κ_{l-1}, then κ_{l-2}, and so on. Assume that the most significant $l - j' - 1$ symbols, namely, $\kappa_{l-1}, \kappa_{l-2}, \cdots, \kappa_{j'+1}$ 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 a two-valued logic. The function's value depends on k; more specifically, for $j = j'$, it depends on $\kappa_{l-1}, \kappa_{l-2}, \cdots, \kappa_{j'}$. The true value, which is still unknown to the attacker, is generated within the crypto-system 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

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

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 $\kappa_{j'}$.

Instead of Algorithm 2, if Algorithm 2a or 2b is used, the representation of k will have n symbols, each belonging to the set $\mathcal{S}$. To obtain a non-zero differential signal, the attacker needs to try all symbols of $\mathcal{S}$, in the worst case, and $|\mathcal{S}|/2$ symbols, on average. In the following, we assume that the crypto-system under DPA attack uses Algorithm 2a or 2b.

4.2 Time and Space Requirements

Based on the above discussions, one can divide the task of the DPA attack into two phases. In phase I, the power consumption signals are monitored. The amount of time needed for this purpose is essentially the time to execute the scalar multiplication algorithm t times corresponding to the input points P_0, P_1, $\cdots$, P_{t-1}. In phase II, the power consumption signals are analyzed. Toward this effort, for the attacker's guess $\kappa'_j \in \mathcal{S}$, the values of the partitioning function and the differential signal (i.e., $\gamma'_{i,j}$, and $\delta(j)$) are computed. Given a $\gamma'_{i,j}$, the time needed to generate the corresponding differential signal is equivalent to the addition of t real numbers. Thus, the computation of all the differential signals takes about $\mathcal{O}(nt|\mathcal{S}|)$ additions over $\mathbb{R}$. On the other hand, the computation of $\gamma'_{i,j}$ may require more complicated operations, such as addition of two elliptic curve points. If we denote p as the time needed to evaluate $\gamma'_{i,j}$, then the time to compute all the $\gamma'_{i,j}$'s is $\mathcal{O}(ntp|\mathcal{S}|)$.

The DPA attack also requires memory space to store elliptic curve points and power consumption signals. For the calculation of $\gamma'_{i,j}$'s, the attacker needs to have t elliptic curve points stored in memory where each such point, in the affine coordinate system and in uncompressed form, will require $2n$ bits. Additionally, the value of the differential signal relies on the power consumption signals. These signals may reside in the memory all at a time, if phases I and II of the DPA attack do not overlap. In this case, the corresponding memory space requirement is $\mathcal{O}(nt\eta)$ bits, where η is the number of bits needed to represent a power signal. The attacker may reduce this space requirement to about $\mathcal{O}(t\eta)$ bits by storing only t signals at a time. For example, to attack κ_j, only $S_{0,j}$, $S_{1,j}$, $\cdots$, $S_{t-1,j}$ are needed. These signals can be generated by executing the main loop of the scalar multiplication algorithm $n-j$ times with input points P_0, P_1, $\cdots$, P_{t-1}, respectively. It is not difficult to see that such a reduced space requirement comes with a price of about $\frac{n+1}{2}$ fold increase in the power-consumption-signal monitoring time. The time and space complexities for the DPA attack can be stated as follows.

Remark 4 *Let n, η, p, t and $\mathcal{S}$ be as defined above. Then the DPA attack has a time complexity of $\mathcal{O}(ntp|\mathcal{S}|)$ and its memory space requirement is $\mathcal{O}(nt\eta)$ bits.*

The values of n and $|\mathcal{S}|$ depend on the crypto-system under attack and can not be typically controlled by the attacker. The value of η can be close to the word size (i.e., 32 bits or so) of the processor used by the attacker. If two elliptic curve points are added to evaluate the partitioning function, then p can be approximated as $\mathcal{O}(n^2)$ bit level operations (i.e., logical AND, XOR, etc.). In order to reduce the time and space requirements, the attacker's possible strategy is to reduce t, i.e., to obtain the differential signal of sufficient strength with fewer power consumption signals.

4.3 Improving Differential Signal

The effectiveness of the DPA attacks depends on the attacker's ability to successfully generate the differential signal δ for the correct guess of the key symbols. The attacker attempts to maximize this signal in an effort to simplify the analysis on the monitored power signals. In [20], a number of techniques including noise reduction using digital filtering and signal magnification by multiple bit DPA, have been investigated which can maximize the differential signal. In [8], this function is a specific bit of the intermediate point that the crypto-system processes. The DPA relies on the assumption that there is a difference between power consumptions when that specific reference bit is 0 and 1.

The partitioning function γ plays an important role in determining the value of the differential signal. If we group the power signals into two resultant signals S_0 and S_1, and give equal weights to all individual power signals $S_{i,j}$, then the maximum possible value of the differential signal is

$$\delta_{\max}(j') = \max\{S_{i,j'}, 0 \le i \le t-1\} - \min\{S_{i,j'}, 0 \le i \le t-1\} \tag{13}$$

provided that the crypto-system does not generate any power internally.

To use a simple partition function, the attacker may use only a single bit of the intermediate result of the scalar multiplication. In such cases, instead of choosing a specific bit for all κ_j, $0 \le j \le n-1$, an attacker may strengthen the differential signal by dynamically determining the optimal bit to be used as a partitioning function.

An attacker with plenty of resources will probably attempt to strengthen the differential signal by using a partitioning function as complex as possible. For example, the attacker can choose the number of transitions of logic (from 1 to 0 and vice versa). Certain hardware implementation technologies dissipate considerably uneven amounts of power for these two types of transitions. If the average transition activities for a part of the scalar multiplication algorithm is known, the attacker can use it to split the power signals in two groups which may result in an improved differential signal.

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 crypto-system, 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 provide the same Q.

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

5.1 Key Masking with Localized Operations (KMLO)

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

$$2 = \tau - \tau^2 = -\tau^3 - \tau, \tag{14}$$

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 viz., $(\kappa_{i+3}, \kappa_{i+2}, \kappa_{i+1}, \kappa_i)$ 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

$$\begin{array}{rcccc} & (d_{i+3}, & d_{i+2}, & d_{i+1}, & d_i) \\ = & (0, & \overline{1}, & 1, & \overline{2}) \\ = & (\overline{1}, & 0, & \overline{1}, & \overline{2}) \\ = & (\overline{1}, & 1, & \overline{2}, & 0) \end{array} \tag{15}$$

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 a set that has $\mathcal{S}$ as a subset. Note that for non-overlapping and fixed sized windows, there are a limited number of combinations that are available to mask a window. For a higher level of security the windows can be randomly overlapped and their sizes can be randomly changed. These make the KMLO countermeasure resistant against attacks at the window boundaries.

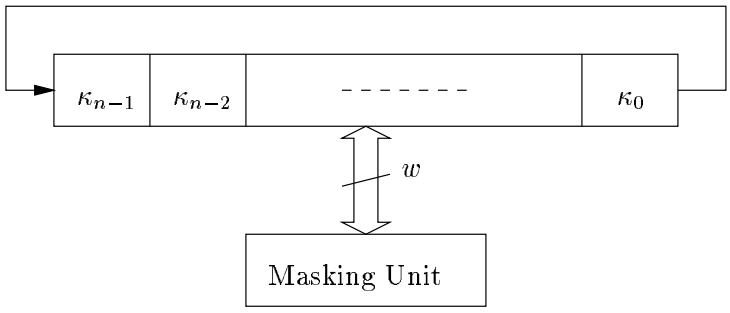

Figure 4: Block diagram of key masking with localized operation (KMLO).

To implement this key masking scheme in hardware, one can use an n-stage shift register where each stage can hold one symbol of $\mathcal{S}$ (see Fig. 4). 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 add it to another vector derived from (14). (For $w = 4$, a set of possible vectors are given in (15).) 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 outside the set $\mathcal{S}$, one can repeatedly apply (14) to restrict the

symbol within $\mathcal{S}$. Additionally, since $(\tau^{n-1} + \tau^{n-2} + \cdots + 1)P \equiv 0P$,

$$Q = \left(\sum_{i=0}^{n-1} \kappa_i \tau^i \right) P \equiv \left(\sum_{i=0}^{n-1} \hat{\kappa}_i \tau^i \right) P \tag{16}$$

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).

Once the key has been masked as described above, the masked key can be used in an SPA resistant algorithm presented earlier to compute the scalar multiplication. Before another scalar multiplication is performed, the masked key stored in the n-stage shift register (refer to Fig. 4) is to be masked again. To prevent a DPA attack, this masking process is repeated before each scalar multiplication.

5.2 Random Rotation of Key (RRK)

Let

$$P' = \tau^r P, \tag{17}$$

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:

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

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: P and $k \equiv (\kappa_{n-1},\ \cdots, \kappa_1,\ \kappa_0)_\tau$
Output: $Q = kP$

```
Pick up a random number r ∈ {0, 1, ···, n − 1}
Compute P' = τ^r P
Q := O; j = r; P'[i] := s_i P',   0 ≤ i ≤ |S| − 1
for (i = 0; i ≤ n − 1; i++) {
    j := j − 1 (mod n)
    Q := τQ
    Q := Q ⊎ P'[κ_j − s_0]
}
```

The above algorithm can be mapped onto a hardware structure as shown below. The secret key is shown to be in a register where each cell can hold any element of $\mathcal{S}$. A simple control unit is assumed to generate the random integer r, which is used by the multiplexor (MUX) to select an appropriate symbol from the register. This symbol is used for accessing the look-up table which holds the elliptic curve points $s_i P'$, $\forall i$. The point retrieved from the table is added to τQ to obtain Q.

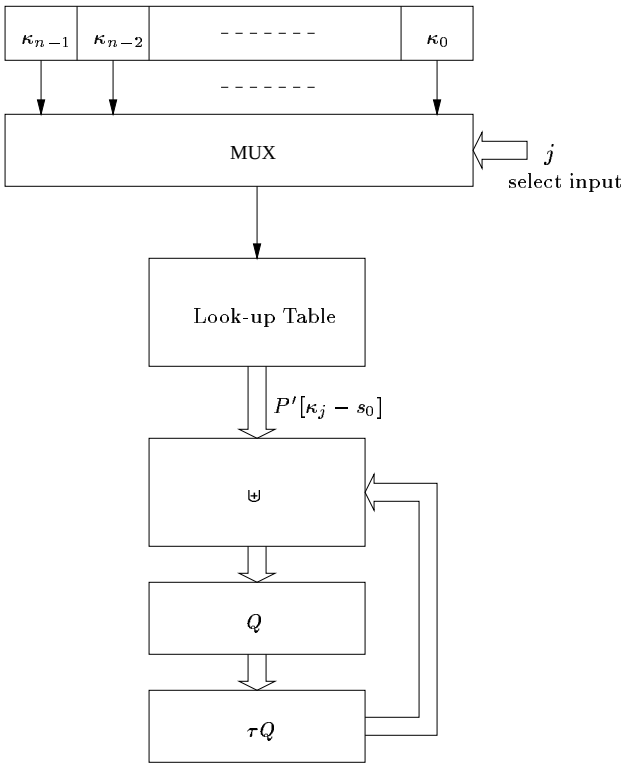

Figure 5: Architecture for elliptic curve scalar multiplication with random rotation of key.

Before starting the iterations, Algorithm 3 computes P'. From (17), we have

$$P' = (x^{2^r}, y^{2^r}) \tag{18}$$

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. Toward this effort, the following steps can be used for computing $P' = \tau^r P$, where $1 \leq r \leq n$ and the function $U(z)$ is equal to 1 if $z \geq 0$, and 0 otherwise.

$P[0] = P$ and $P[1]$= any point on E_α
for $i = 0$ to $n - 1$ {
    $e := U(i - r)$
    $P[e] := \tau P[e]$

}
$P' := P[0]$

In the above procedure, a method to determine $e := U(i-r)$ in an SPA safe manner may be needed. For this purpose, one can use a $\lceil \log_2 n \rceil$ bit 2's complement adder/subtracter to subtract r from i. The most significant bit of the subtraction operation is 0 if $i - r \geq 0$, otherwise 1. Thus, a logical NOT of this bit results in e.

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. Based on this idea, below we present a simple key randomization scheme, which however can be extended to derive more advanced schemes that use redundant symbols.

Before a particular scalar multiplication, assume that a total of n' redundant symbols denoted as f_i, for $0 \leq i \leq n'-1$, are inserted into the original key sequence $\{\kappa_i\}$, for $0 \leq i \leq n-1$. Thus the resultant sequence, denoted as $b = \{b_i\}$, for $0 \leq i \leq N-1$, 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 $n' \geq 4$, if the pair (f_{2l}, f_{2l+1}), for $0 \leq l < n'/2 - 1$, 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 $N - 1 \geq i > j > 1$.

In order to implement this scheme, an N bit sequence g is generated which has $\frac{n'}{2}$ non-adjacent one's. The locations of these 1's are random and 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.

$$\begin{array}{lccccccc} \text{Location}: & N-1 & \cdots & N-r+1 & N-r & N-r-1 & N-r-2 & \cdots \\ g: & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots \\ b: & \kappa_{n-1} & \cdots & \kappa_{n-r+1} & f_0 & f_1 & \kappa_{n-r} & \cdots \end{array} \tag{19}$$

In (19), the first pair (f_0, f_1) has been inserted at random location $N - r$. Assume that all b_i's belong to an *expanded* symbol set $\mathcal{U} = \{u_0, u_1, \cdots, u_{|\mathcal{U}|-1}\} \supseteq \mathcal{S}$, where u_i's may not be integers. 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,|$\mathcal{U}$|-1] and is obtained as $b'_i = \imath$, for $0 \leq \imath \leq |\mathcal{U}| - 1$, given that $b_i = u_\imath$.

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

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

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 (19). To obtain the correct Q at the end of iteration $i = N - r + 2$, the algorithm should yield

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

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:

$$\begin{aligned} Q^{(N-r-1)} &= \tau^{-1}(\kappa_{n-1}\tau^{r-1} + \kappa_{n-2}\tau^{r-2} + \cdots + \kappa_{n-r+1}\tau + f_0) + f_1, \\ Q^{(N-r-2)} &= \kappa_{n-1}\tau^{r-1} + \kappa_{n-2}\tau^{r-2} + \cdots + \kappa_{n-r+1}\tau + f_0 + f_1\tau + \kappa_{n-r}. \end{aligned} \quad (21)$$

Comparing (20) and (21), 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

$$f_0 + \tau f_1 = 0. \quad (22)$$

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

$$f_2 + \tau f_3 + \tau^{m-1}(f_0 + \tau f_1) = 0 \quad (23)$$

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.

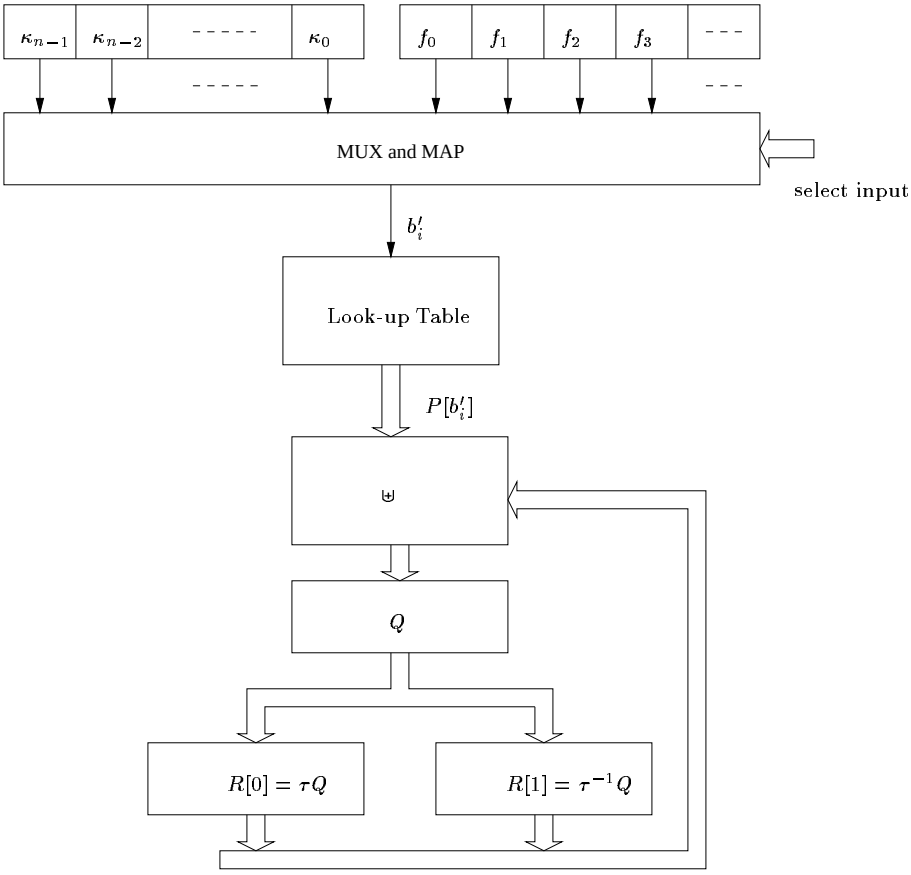

Figure 6: Architecture for elliptic curve scalar multiplication with random insertion of redundant symbols.

A hardware structure for the implementation of Algorithm 4 is shown in Fig. 6. The loop of the above algorithm requires two extra elliptic curve points to be stored. It also requires the inverse of the τ mapping, which is however simple.

5.4 Discussions

Here we first make a few comments on the countermeasures proposed above. Then we discuss the costs to implement these countermeasures.

5.4.1 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 and hence potentially to other crypto-systems.
- Each of the three proposed methods uses a look-up table which hold the pre-computed points. 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 around zero, then the look-up table size can be reduced. 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 can reduce the table size up to 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.
- Like the SPA countermeasures, the computational complexities of the DPA countermeasures presented above are not affected by the number of non-zero symbols in the reduced τ-adic representation of k.

5.4.2 Overheads

The DPA countermeasures presented earlier use key randomization in conjunction with SPA resistant scalar multiplication. Below we discuss the main costs due to these key randomization schemes.

- In the KMLO scheme, the key is masked w symbols at a time (refer to Fig. 4). The masking vectors, which are given in (15) for $w = 4$, can be stored in a small look-up table of w symbols wide. For today's resource constrained crypto-systems, each symbol is expected to be of about three bits or so. The KMLO scheme also requires w adders each capable of adding any two symbols used for the representation of k. Thus, the time complexity of this key randomization scheme can be approximated as $\mathcal{O}(\frac{n}{w})$. Simple trade-offs between space and time complexities are possible, such as to use only one adder and have a time complexity of $\mathcal{O}(n)$.
- In the RRK scheme, the key k is given a r fold cyclic shift, where r is a random integer in the range $[0, n-1]$. For this purpose, a $\lceil \log_2 n \rceil$ bit random number generator can be used. If a normal basis is used for representing the elements of GF(2^n), then RRK does not require any more additional resources. When a polynomial basis is used, one would however need $2n$ extra squaring operations in GF(2^n) corresponding to the n Frobenius mappings used for the secure computation of $P' = \tau^r P$.
- The random locations of the redundant symbols used in the RIRS scheme can be generated with a $\lceil \log_2 n \rceil$ bit random number generator. A look-up table is used to store the pre-computed points. This table is similar to the table needed for Algorithm 2b. The look-up table shown in Fig. 6 is however larger by an amount equivalent to the storage space of $|\mathcal{U}| - |\mathcal{S}|$ elliptic curve points, where $\mathcal{U}$ and $\mathcal{S}$ are the symbol sets discussed in Section 5.3. The value of $|\mathcal{U}| - |\mathcal{S}|$ is less than n', and is equal to zero when the redundant symbols belong to $\mathcal{S}$. As shown in Fig. 6, the other overheads for the RIRS scheme include storage space for the redundant symbols, a multiplexer and an inverse Frobenius mapping unit. The inverse of the Frobenius map corresponds to two square root operations in GF(2^m) and has space and time complexities comparable to the Frobenius map. Overall, the computational overhead for Algorithm 4 is about $(1 + \frac{n'}{n})$ times that of Algorithm 2b.

6 Impact on Differential Signal

In this section, we discuss the effect of the above three countermeasures on the differential signal of the DPA attack. For the sake of clarity, below we write ϵ of (12) as ϵ_∞ since it corresponds to the differential signal value as $t \to \infty$. Now, let $\epsilon_{\min}$ be the minimum value of the differential signal needed by the attacker to distinguish $\kappa'_{j'} \neq \kappa_{j'}$ from $\kappa'_{j'} = \kappa_{j'}$ (refer to (12)). The attacker can obtain this $\epsilon_{\min}$ by running the scalar multiplication operations (and monitoring the power signals), say only $t_{\min}$ times. The latter is most likely to be much less than his maximum ability of t_{able} runs. Let us denote ϵ_{able} as the value of the differential signal corresponding to t_{able}, such that $\epsilon_{\min} \leq \epsilon_{\text{able}} \leq \epsilon_\infty$.

When the scalar multiplication operations are equipped with DPA countermeasures, assume that the attacker uses his ability of t_{able} runs to recover the scalar. Let ϵ_A, ϵ_B and ϵ_C denote the values of the differential signals that the attacker obtains from his DPA attacks on the scalar multiplication operations equipped with countermeasures KMLO, RRK and RIRS, respectively, described earlier.

Since $\kappa \equiv k \bmod (\tau^n - 1)$, the KMLO countermeasure is essentially a mapping of the binary scalar to one of many τ-adic equivalent scalars. The resultant scalars use a symbol set of size $|\mathcal{S}|$ and are of n symbols each. We assume that the windows for the masking can be overlapped and their sizes can be varied to various sizes, so that the total number of τ-adic n-tuples is $(|\mathcal{S}|)^n$. Then, each binary scalar can take, on average, $(|\mathcal{S}|/2)^n$ equivalent τ-adic scalars. If the masking operation is random, the differential signal is attenuated by a factor of $(|\mathcal{S}|/2)^n$, i.e.,

$$\epsilon_A \approx \frac{\epsilon_{\text{able}}}{(|\mathcal{S}|/2)^n}.$$

In RRK, the scalar multiplication operation starts from a random symbol position of τ-adic representation of κ. Assuming that the starting position is uniformly distributed over all n positions, the attacker's differential signal is attenuated by a factor of n, i.e.,

$$\epsilon_B \approx \epsilon_{\text{able}}/n.$$

As mentioned earlier, the RIRS countermeasure can be implemented at various levels of complexities and these would affect the level of protection against DPA attacks as well as the implementation cost and computation time. Here we consider a simple version of RIRS based on (23) with $m = 1$. Thus, four redundant symbols are grouped together and n' is a multiple of four. In each group, the effect of the first pair is cancelled by the second pair. We pick the first pair from the symbol set $\mathcal{S}$. In order to satisfy (23), the second pair would belong to an extended set $\mathcal{U}$ where $|\mathcal{S}| \leq |\mathcal{U}|$. If there is at least one τ-adic symbol of κ in between two groups of redundant symbols, then $n'/4 \leq n + 1$, and each scalar is mapped onto one of the $\binom{n+1}{n'/4} (|\mathcal{S}|)^2$ combinations and so is the factor of attenuation of the differential signal, and we have

$$\epsilon_C \approx \frac{\epsilon_{\text{able}}}{\binom{n+1}{n'/4} (|\mathcal{S}|)^2}.$$

In the above discussions, the effects of the three countermeasures are considered separately. When combined they contribute to the randomization of the scalar by reducing the differential signal by their respective attenuating factor. Thus, if all the three countermeasures are applied together, the differential signal is attenuated by a factor which is equal to the product of the above three attenuation factors, i.e.,

$$\frac{n}{4}(|\mathcal{S}|/2)^{n+2} \binom{n+1}{n'/4} (|\mathcal{S}|)^2.$$

For practical crypto-systems based on Koblitz curves, one would find $n \geq 163$ and $|\mathcal{S}| \approx 3$. For these values of n and $|\mathcal{S}|$, the differential signal is attenuated to a level which makes the DPA attack infeasible with technologies available today.

7 Comparison

In this section, we briefly compare a number of DPA countermeasures, namely [8], [7] and [21], with those presented in this article. These countermeasures are of similar type in the sense that their resistance depends on the randomization of the secret key.

In the randomization technique of [8], a multiple of the total number of curve points $\mathcal{E}$ is added to k. Since $\mathcal{E}P = \mathcal{O}$,

$$Q = kP = (k + e\mathcal{E})P \tag{24}$$

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 [7] 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 a relatively more complex implementation.

Although, the method proposed in [21] 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 crypto-systems, 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 a random position (i.e., symbol), but unlike [21] it terminates adjacent to the random starting position. As a result, it is less vulnerable to the backward power analysis attack.

8 Concluding Remarks

The security of many public-key crypto-systems depends on the secrecy of the private key. In the recent past, a number of methods have been reported in the open literature to recover the key by analyzing the power consumption of a crypto-system. If an adversary is successful in obtaining the key, he can impersonate the actual key holder and commit malicious acts. In many applications, the key is stored inside the crypto-system and appropriate measures are to be taken so that the key is not revealed when a power analysis attack is launched.

This article has considered power analysis attacks on Koblitz curve based crypto-systems. Algorithmic level countermeasures have been presented for both the SPA and DPA attacks. The proposed SPA resistant schemes rely on making the power consumption for the scalar multiplication independent of the values of the key. Since these schemes also make all the instructions of the scalar multiplication algorithm run irrespective of the value of the key, they can be potentially extended to the protection against timing attacks.

The DPA resistant schemes proposed in this article rely on the randomization of the key as follows: (i) key masking without a large multiplier, (ii) random rotation of the key, and (iii) random insertion of redundant symbols in the key. The first scheme reduces the implementation complexity of DPA resistant design by eliminating the need for a very large integer multiplier. Consequently, it also reduces time delays due to carry-propagation inherent to many large multipliers. The second scheme randomly rotates the symbols of the key. The third scheme is the insertion of redundant symbols at random locations of the key making power analysis attacks combined with pattern search infeasible.

When applied to Koblitz curve based crypto-systems, the above countermeasures are expected to be less complex than the similar ones proposed recently in the open literature. Nevertheless, their overall impacts on the crypto-systems 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

Most of the work was 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] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4:3–72, 1991.

[2] E. Biham and A. Shamir. Differential Cryptanalysis of the Full 16-round DES. In *Advances in Cryptology- CRYPTO '92, Lecture Notes in Computer Science*, pages 487–496. Springer-Verlag, 1993.

[3] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology- CRYPTO '97, Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.

[4] I. F. Blake, G. Seroussi, and Nigel P. Smart. *Elliptic Curves in Cryptography.* Cambridge Univ Press, 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*, pages 37–51. Springer-Verlag, 1997.

[6] Certicom Research. *GEC2: Recommended Elliptic Curve Domain Parameters.* Standards for Efficient Cryptography Group, http://www.secg.org, September 2000.

[7] S. Chari, J. R. Rao C. S. Jutla, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Advances in Cryptology- CRYPTO '99*, pages 398–412, 1999.

[8] Jean-Sebastien Coron. Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems. In *Workshop on Cryptographic Hardware and Embedded Systems*, pages 292–302. LNCS, Springer-Verlag, 1999.

[9] D. Gordon. A Survey of Fast Exponentiation Methods. *Journal of Algorithms*, 27:129–146, 1998.

[10] M. A. Hasan. Power Analysis Attacks And Algorithmic Approaches To Their Countermeasures For Koblitz Curve Crypto-system. In *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, pages 94–109. LNCS, Springer-Verlag, 2000.

[11] J. Kelsey. Side Channel Cryptanalysis of Product Ciphers. In *ESORICS, Lecture Notes in Computer Science*, pages 487–496. Springer-Verlag, 1998.

[12] 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*, pages 176–189. Springer-Verlag, 1999.

[13] N. Koblitz. CM-Curves with Good Cryptographic Properties. In *Advances in Cryptology- CRYPTO '91, Lecture Notes in Computer Science*, pages 279–287. Springer-Verlag, 1992.

[14] N. Koblitz. Elliptic Curve Cryptosystems. *Math. Comp.*, 48:203–209, 1993.

[15] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Systems. In *Advances in Cryptology- CRYPTO '96, Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[16] P. Kocher, J. Jaffe, and B. Jun. Introduction to Differential Power Analysis and Related Attacks. http://www.cryptography.com/dpa/technical, 1998.

[17] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology- CRYPTO '99, Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[18] M. Matsui. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology- EUROCRYPT '93, Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994.

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

[20] T. Messerges, E. A. Dabbish, and R. H. Sloan. Investigation of Power Analysis Attacks on Smartcards. In *Proceedings of USENIX Workshop on Electronic Commerce*, pages 151–161, 1999.

[21] 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*, pages 144–157. LNCS, Springer-Verlag, 1999.

[22] V. S. Miller. Use of Elliptic Curves in Cryptography. In *Advances in Cryptology- CRYPTO '85*, pages 417–426. Springer, 1986.

[23] R. Schroeppel, S. O'Malley, H. Orman, and O. Spatscheck. Fast Key Exchange with Elliptic Curve Systems. In *Advances in Cryptology- CRYPTO '95, Lecture Notes in Computer Science 963*, pages 43–56. Springer-Verlag, 1995.

[24] J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106. Springer-Verlag, New York, 1986.

[25] J. Solinas. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In *Advances in Cryptology- CRYPTO '97, Lecture Notes in Computer Science*, pages 357–371. Springer-Verlag, 1997.

[26] U.S. Department of Commerce/NIST. *Digital Signature Standards (DSS)*. Federal Information Processing Standards Publications, http://csrc.nist.gov/cryptval, January 2000.

[27] H. Wu. Low Complexity Bit-Parallel Finite Field Arithmetic Using Polynomial Basis. In *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, pages 280–291. LNCS, Springer-Verlag, 1999.

[28] 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*, pages 269–279. LNCS, Springer-Verlag, 1999.