# SCA Countermeasures for ECC over Binary Fields on a VLIW DSP Core

E. Smith* and C. Gebotys**

*IDERS INCORPORATED,
Winnipeg MB Canada,
esmith@iders.ca

**Dept of Elect. & Comp. Engineering
University of Waterloo, Waterloo ON Canada,
cgebotys@optimal.vlsi.uwaterloo.ca

**Abstract.** In recent years, Kocher introduced SCA techniques to the cryptographic community. Contrary to previous cryptanalysis methods that attack the mathematically difficult problems cryptographic techniques are based on, SCAs exploit physical properties of implementations in an attempt to compromise systems. Following the introduction of the new cryptanalysis techniques, numerous algorithms have been proposed that reduce or eliminate their effectiveness. Focusing on ECC, the paper proposes numerous SPA and DPA countermeasures relevant to random and Koblitz curve implementations. The countermeasures are described and briefly analyzed, including stating how they are proposed to reduce the effectiveness of the attacks. The implementation and performance of the countermeasures on a specific DSP is described. Furthermore, power traces of implementations of the techniques are examined for SPA attempts, as well as investigating the effectiveness of simulated DPA attacks on the DSP.

## 1 Introduction

Recently, Kocher described cryptanalysis techniques based on side-channel information, including power consumption and timing data. Side Channel Attacks (SCAs) exploit the power consumption of a processor and timing information in an attempt to compromise cryptographic implementations. The foremost power consumption based attacks are Simple Power Attacks (SPAs) and Differential Power Analysis (DPA). SPAs exploit algorithms whose execution sequence depends on pertinent information. DPA is a more sophisticated attack than SPA, which is based on a statistical analysis of multiple power traces. In particular, it utilizes the correlation between specific points within several power traces and bits of pertinent data.

With respect to Elliptic Curve Cryptography (ECC), Coron describes SCA techniques to recover the key k, involved in the point multiplication operation [2]. Coron presents an SPA resistant point multiplication operation, as well as DPA countermeasures. SPA attempts are relevant to all elliptic curve based cryptosystems, including both key exchange, encryption and digital signature techniques, whereas DPA attacks are only relevant to the first two techniques because of the nonce involved in elliptic curve digital signature techniques [2].

In recent years, a significant amount of research has focused on SCAs and cryptography. Several articles referenced throughout the paper present various SCA countermeasures. Most DPA countermeasures are implementation specific, meaning that the countermeasure only applies to target systems employing projective coordinates or a Koblitz curve. DPA countermeasures that apply to all elliptic curves, and some that only apply to Koblitz curves are presented later. Koblitz curves are beneficial for implementation because of the associated $\tau$-adic representation of polynomials, which greatly improves the performance of the point multiplication operation. See [19] for a thorough description.

Within the paper, implementation of the countermeasures is done using a Koblitz curve, but this is not necessary in all cases. Furthermore, unlike most of the previous ECC research that targets general-purpose processors, the implementation of the proposed countermeasures is done on a DSP, more specifically the Motorola StarCore SC140 DSP (SC140). The processor is a high performance DSP primarily targeting wireless and wireline high-bandwidth communications [16]. It can operate at a maximum of 300 MHz, and has four parallel operating ALUs, each capable of executing a multiply-and-accumulate per clock cycle.

SPA and a technique to resist the attack are presented in §2. DPA, and several proposed countermeasures for Koblitz and non-Koblitz curves are described in §3. The performance of the countermeasures is presented in §4, followed by power traces of the implemented point multiplications that are briefly analyzed in §5. Lastly, the results of several simulated DPA attempts are presented in §6.

## 2 Simple Power Attacks

SPA is the simplest SCA, where the power trace is directly analyzed to determine information pertinent to the cryptosystem. Pertinent information refers to key bits, or other data that can compromise the cryptosystem in question. When attempting an SPA, an attacker determines the sequence of executed instructions by analyzing the power trace of the processor. The attackers can then determine pertinent information about the cryptosystem if the sequence of executed instructions is dependant. To resist SPA attempts, algorithms must be written such that the instructions executed do not depend on pertinent information. For example, within the loop of the commonly used elliptic curve point multiplication operation, a point addition operation is only executed if the corresponding bit is set. The algorithm is susceptible to SPA attempts because the execution sequence depends directly on the value of the large integer *k* involved in the point multiplication operation, where k is pertinent information that must be kept secret. By analyzing the resulting power trace, an attacker can easily determine if a point addition operation was performed within a specific loop iteration, and therefore can determine the value of k. A simple modification to the algorithm was proposed by Coron [2] to ensure the instructions executed are independent of the value of k. The SPA resistant point multiplication algorithm is presented as Algorithm 1. The symbol $\mathcal{O}$ represents the point at infinity.

**Algorithm 1.** SPA Resistant Point Multiplication [2]

Input: $k = \{k_{m-1}, k_{m-2}, \ldots, k_1, k_0\}_2$, P
Output: Q
1. $Q = \mathcal{O}$
2. For i = m-1 to 0 do
   2.1. temp[0] = 2·Q
   2.2. temp[1] = P + temp[0]
   2.3. $Q = temp[k_i]$
3. Return(Q)

In the above algorithm, a point addition is computed every loop iteration, independent of the value of k. Therefore, an attacker is unable to determine the value of k from a single power trace. The more complex DPA attack, which uses a statistical analysis of several power traces, is required to determine k.

## 3 Differential Power Analysis

Generalized to elliptic curve cryptosystems by Coron [2], DPA attacks are much more sophisticated than SPA. In a DPA attack, an attacker statistically analyzes several power traces to strengthen the differential signal present in each power trace 6. By strengthening the differential signal, the correlation between specific points of the differential power trace and bits of pertinent information is increased. Thus, an attacker is able to determine pertinent bit values and compromise the security of the cryptosystem. Detailed descriptions of DPA theory and numerous countermeasures to foil attacks are presented in [1], [2], [4], [6], [7], [9], [11] and [18].

In general, DPA countermeasures add some type of randomness to the point multiplication operation in an attempt to reduce or eliminate the correlation between power traces and pertinent information. DPA countermeasures are usually implementation specific, more complicated than Algorithm 1, and generally exploit advantageous properties of the implementation or specific elliptic curve.

For example, projective coordinate countermeasures are mentioned in [3], [4], [15] and [18]. The countermeasure does not apply to implementations that employ the affine coordinate system. Furthermore, Hasan, Joye, and Tymen present several DPA countermeasures that exploit Koblitz curve properties in [6] and [9] respectively, which do not apply to cryptosystems employing non-Koblitz curves. In the following sections, several proposed DPA countermeasures are offered, along with brief analysis. The actual unpredictability of the random data generated in each proposed countermeasure determines the effectiveness of each algorithm in resisting DPA attacks. In each countermeasure, random data and values directly related to k, such as k', must be computed in a secure manner, which is not susceptible to attacks.

### 3.1 Proposed Koblitz Curve Specific Countermeasures

Koblitz curve implementations are attractive because they increase the performance of the point multiplication operation. By using $\tau$-adic representations of k in the elliptic curve point multiplication operation, point doubling operations are replaced with two finite field squaring operations, resulting in a significant reduction in computational costs. Similar to countermeasures presented in [6], the below countermeasures exploit a property of Koblitz curves, and are proposed to reduce the effectiveness of DPA attacks. The property, which is also known as the Frobenius mapping, is presented as Formula 1. It describes the inexpensiveness of multiplying elliptic curve points by $\tau$.

$$\tau \cdot (x, y) := (x^2, y^2) \tag{1}$$

Each of the Koblitz curve specific countermeasures listed in this section can be generalized to apply to non-Koblitz curve implementations, however, Formula 1 can no longer be exploited. Therefore, the computational overhead associated with the countermeasures is extremely large, making implementation unfeasible.

In each of the countermeasures, a $\tau$-adic representation of the large integer k is required. It is suggested that Non-Adjacent Format (NAF) related representations of k be avoided, otherwise the distribution of coefficients favor zero, which may lead to reduced DPA resiliency. Furthermore, when employing the countermeasures, SPA attempts must be avoided; eliminating the advantage of employing a NAF related representation.

#### 3.1.1 Proposed Koblitz Curve Specific Countermeasure 1

The first Koblitz curve specific countermeasure is loosely based on a finite field exponentiation algorithm. The countermeasure is also similar to an exponent splitting algorithm presented by Clavier and Joye in [2]. The $\tau$-adic representation of k is divided into r groups of g coefficients, where $g = \lceil m/r \rceil$. The point multiplication, between each group and base point P, is performed in a random order. Then, the resulting points are multiplied by the corresponding power of $\tau$, and summed, resulting in the point Q. The countermeasure is formally presented below as Algorithm 2.

**Algorithm 2.** Koblitz Curve Countermeasure 1

Input: $k = (k_{m-1}, k_{m-2}, \ldots, k_1, k_0)_\tau$, P
Output: $Q = k \cdot P$
1. Compute $k^i$, for $0 \le i \le r-1$
   Where $k = (k^{r-1}, k^{r-2}, \ldots, k^1, k^0)_\tau$ and $k^i = (k_{(i+1)\cdot g-1}, k_{(i+1)\cdot g-2}, \ldots, k_{i\cdot g+1}, k_{i\cdot g})_\tau$
2. Randomly order b, where $b = \{0, 1, \ldots, r-1\}$
3. For i = 0 to r-1

3.1. Compute $Q^{b(i)} = k^{b(i)} \cdot P$, where b(i) is the $i^{th}$ element of b.
4. Compute $Q = \Sigma \tau^{i \cdot g} \cdot Q^i$, for $0 \leq i \leq r-1$

The crucial step in the countermeasure is computing $k^i \cdot P$ (step 3.1) following the random sequence b. It is speculated that this step adds randomness to the power trace of the algorithm, which reduces the effectiveness of DPA attacks. To further analyze the proposed countermeasure, a concept of Sample Entropy (SE) is introduced.

As DPA attacks mature, requiring fewer samples to be effective, an idea of SE, or uncertainty per sample, which refers to the amount of randomness introduced by a countermeasure to the power trace, may become important when analyzing the effectiveness of DPA countermeasures. In the case of point multiplication, SE can be thought of as the number of variations of the operation that are possible using a single algorithm to compute k·P. For example, the straightforward double-and-add algorithm has an SE of one, and therefore is very susceptible to SPA and DPA. Algorithms with SE values larger than one are not susceptible to SPA, and as their associated SE increases, an algorithm becomes less susceptible to DPA attempts. The amount of entropy introduced to an attacker monitoring a cryptosystem and the amount of overhead associated with a countermeasure are important, both of which are controllable in Algorithm 2. The SE, as well as the overhead of the algorithm vary with r, and are estimated in Table 1. The overhead of Algorithm 2, excluding the point multiplication operations in step 3.1, is reasonable for implementations that do not have strict memory restrictions. The memory overhead cannot be reduced by combining steps 3 and 4 of the algorithm, because that would introduce DPA vulnerabilities. In the tables presented throughout the paper, PA, SQ and FF represent point additions, finite field squarings and finite field elements respectively. In addition, the memory overhead of a point is considered equal to two finite field elements.

Modifications can be made to Algorithm 2 to increase its performance. For example, a windowing technique can be exploited in step 3.1. In this case, the Look-Up Table (LUT) is computed once and used in all point multiplication operations, reducing the overall computational cost.

Algorithm 2 is a proposed DPA countermeasure. Steps must be taken to resist SPA attempts as well. A resistant point multiplication algorithm, such as Algorithm 1, is required in step 3.1. Employing large r-values increases the SE, but large r-values cannot be used to resist SPA attempts and lead to unacceptable computational and memory overheads.

**Table 1.** Estimated Sample Entropy and Overhead of Algorithm 2

| Sample Entropy | | Overhead | |
|---|---|---|---|
| **Small r** $P(k^i \text{ coll}) = 0$ | **Large r** $P(k^i \text{ coll}) = 1$ | **Operation Count** | **Estimated Memory** |
| r! | $\frac{r!}{((r/2 \cdot g)!)2^g}$ | $(r-1) \cdot PA + (2 \cdot g \cdot (r-1)) \cdot SQ$ | $(2 \cdot r) \cdot FF$ |

In Algorithm 2, the security and ability to thwart DPA attacks is transferred to step 2, where the random order in which $Q^{b(i)} = k^{b(i)} \cdot P$ is computed. Assuming that step 2 is implemented in a secure manner that adds sufficient randomness to b, it is proposed that Algorithm 2 lessens the effectiveness of DPA attacks. The correlation between bits of k and power traces is reduced.

A weakness of Algorithm 2 is that its DPA resiliency depends on the randomness of the large integer k involved in the point multiplication. For example, consider an instance when k is not sufficiently random. Bit patterns within k, whose length divides g, reduce the effectiveness of the countermeasure. Bit patterns reduce the actual amount of SE, resulting in a greater correlation that is exploitable by DPA attacks.

### 3.1.2 Proposed Koblitz Curve Specific Countermeasure 2

The second countermeasure presented in the paper is proposed to resist SPA and counteract DPA attacks, also exploiting Formula 1. Within each loop iteration, two points are multiplied by the radix, $\tau$, which is an inexpensive task. In effect, the countermeasure presented below as Algorithm 3, computes both k·P and

k'·P, where k' is the logical complement of k. It is proposed that SPA attempts are resisted by the use of a nonce, k and k'. By utilizing the logical compliment of k, it is guaranteed that one point addition operation is computed per loop iteration.

**Algorithm 3.** Koblitz Curve Countermeasure 2

Input: $k = \{k_{n-1}, k_{n-2}, \ldots, k_1, k_0\}_\tau$, P
Output: Q = k·P
1. Generate the nonce j, a random (m+1)-bit sequence
2. $kval[j_m] = k$, $kval[j_m'] = k'$
3. Q[1] = $\mathcal{O}$, Q[0] = $\mathcal{O}$
4. For i = m-1 to 0 do
    4.1. $Q[j_i] = \tau \cdot Q[j_i]$
    4.2. $bit = kval_i[j_i]$
    4.3. if (bit = 1) $Q[j_i] = Q[j_i] + P$
    4.4. $Q[j_i'] = \tau \cdot Q[j_i']$
    4.5. $bit = kval_i[j_i']$
    4.6. if (bit = 1) $Q[j_i'] = Q[j_i'] + P$
5. Return($Q[j_m]$)

Algorithm 3 utilizes a nonce to resist SPA and lessen the effectiveness of DPA attacks. The (m+1)-bit nonce j is used to randomize the elements of Q containing the result of k·P and k'·P, as well as the order in which calculations relating to k and k' are performed. The SPA resiliency of the algorithm depends on the size of the nonce, and is reduced if nonce bit values are reused deterministically. SPA attempts are resisted because an attacker will be unable to distinguish between the set of steps, either steps 4.1-4.3 or 4.4-4.6, which involve k. DPA attempts are lessened because the nonce j randomizes both the location of Q=k·P and the order in which calculations relating to k and k' are performed. The spikes observed using a DPA on Algorithm 3 and the straightforward implementation of point multiplication are half as large.

It is possible to improve the performance of Algorithm 3 by utilizing a windowing technique as presented in Algorithm 4. In the algorithm, the size of the nonce j is reduced by a factor of the window width, w, and there are two point additions per loop iteration. An SPA resistant implementation of the point addition operation should be employed in steps 4.3 and 4.6 of Algorithm 3, where a dummy point addition operation is performed when one or both of the point(s) involved is $\mathcal{O}$. This is required to maintain full SPA resiliency. The computational overhead of the additional point addition operation per loop iteration reduces the benefits of employing a LUT, but as shown in §4, the computational cost of Algorithm 4 is still less than Algorithm 3. As stated with Algorithm 3, the SPA resiliency of Algorithm 4 is reduced if bit values of the nonce are reused deterministically.

**Algorithm 4.** Koblitz Curve Countermeasure 2 with Windowing

Input: $k = \{k_{n-1}, k_{n-2}, \ldots, k_1, k_0\}_\tau$, P, w
Output: Q = k·P
1. Generate the nonce j, a random ($\lceil m/w \rceil$+1)-bit sequence
2. $kval[j_{\lceil m/w \rceil}] = k$, $kval[j_{\lceil m/w \rceil}'] = k'$
3. Compute $Pval[i] = i \cdot P$, where $0 \le i \le 2^w - 1$
4. Q[1] = $\mathcal{O}$, Q[0] = $\mathcal{O}$
5. For i = $\lceil m/w \rceil$-1 to 0 do
    5.1. $T = \tau^w \cdot Q[j_i]$
    5.2. Compute coeff (from $kval[j_i]$)
    5.3. $Q[j_i] = T + Pval[coeff]$
    5.4. $T = \tau^w \cdot Q[j_i']$
    5.5. Compute coeff (from $kval[j_i']$)
    5.6. $Q[j_i'] = T + Pval[coeff]$
6. Return($Q[j_{\lceil m/w \rceil}]$)

As previously stated, SPA attempts are resisted in Algorithm 4 if a point addition operation is implemented that performs dummy point addition operations when one or both of the points involved in steps 4.3 and 4.6 are $\mathcal{O}$. Within each algorithm, j and k' must be computed in a secure manner, which is not susceptible to attacks. Without these stipulations, an external party could attack the computation of j, exploit the unequal probability of $j_i$, or determine k from k'. An entity may then be able to successfully compromise the implementation.

The computational and memory overhead for both Algorithm 3 and Algorithm 4 is minimal. Both algorithms require twice as many finite field squaring operations, as well as a slight increase in memory to store j, k' and the result of k'·P. The overhead of both algorithms is relatively small.

To improve the DPA immunity of an implementation, the randomized table method presented by Itoh [8] can be added to Algorithm 3 or Algorithm 4. The DPA resiliency of the resulting point multiplication operation is further increased.

### 3.1.3 Exponent Splitting Variation

The overhead associated with the exponent splitting DPA countermeasure presented in [1] is generally deemed unacceptable. The overhead associated with the countermeasure is reduced by slightly modifying their technique and combining it with techniques outlined in this paper. Without employing a windowing method, the proposed countermeasure would require the nonce j and k'. In Algorithm 5, $k^1 \cdot P$ and $k^2 \cdot P$ are computed simultaneously, where $k=(k^1 + k^2)=(k^1 \mid k^2)$. The symbol | represents the logical 'OR' operator. Bit values of k' are used in dummy point addition operations to ensure a point addition operation is performed every loop iteration. Furthermore, similar to the two previous algorithms, j is used to randomize the order in which $k^1$, $k^2$ and k' are involved in operations.

The performance of the algorithm described in the previous paragraph is greatly improved by employing a windowing technique. The resulting algorithm, presented as Algorithm 5, is very similar to Algorithm 4, with the exception that it does not require k'. Furthermore, $k=(k^1 \mid k^2)$ is not required, as in the description of the countermeasure in the previous paragraph. Only the stipulation $k=(k^1 + k^2)$ is required.

**Algorithm 5.** Exponent Splitting Variation (Koblitz curve specific with windowing)

Input: $k = \{k_{n-1}, k_{n-2}, \ldots, k_1, k_0\}_\tau$, P, w
Output: $Q = k \cdot P$

1. Generate the nonce j, a random $(\lceil m/w \rceil+1)$-bit sequence
2. Generate $k^1$ and $k^2$, random m-bit large integers such that $k = (k^1 + k^2)$
3. $kval[j_{\lceil m/w \rceil}] = k^1$, $kval[j_{\lceil m/w \rceil}'] = k^2$
4. Compute $Pval[i] = i \cdot P$, where $0 \le i \le 2^w-1$
5. $Q[1] = \mathcal{O}$, $Q[0] = \mathcal{O}$
6. For $i = \lceil m/w \rceil-1$ to 0 do
   - 6.1. $T = \tau^w \cdot Q[j_i]$
   - 6.2. Compute coeff (from $kval[j_i]$)
   - 6.3. $Q[j_i] = T + Pval[coeff]$
   - 6.4. $T = \tau^w \cdot Q[j_i']$
   - 6.5. Compute coeff (from $kval[j_i']$)
   - 6.6. $Q[j_i'] = T + Pval[coeff]$
7. Return(Q[0] + Q[1])

Similar to Algorithm 4, to maintain full SPA resiliency, an SPA resistant point addition operation is required in steps 6.3 and 6.6, where a dummy point addition operation is performed when one or both point(s) involved is $\mathcal{O}$. The effectiveness of the above algorithm to lessen DPA attacks is the same as that of Algorithm 4, and it also includes the desired characteristics of exponent splitting as described in [1] and [2].

Algorithm 5 is an extension of the exponent splitting technique proposed in [2]. It extends the algorithm to apply to point multiplication over elliptic curves, as well as using a nonce and windowing technique to improve SPA resiliency and reduce computational costs respectively.

### 3.2 Proposed General DPA Countermeasures

The following sections illustrate DPA countermeasures that apply to any elliptic curve. They do not depend on specific implementation details other than the commonly used multiply by radix and add technique of performing elliptic curve point multiplication operations.

The previous algorithms require two point multiplication operations involving the radix $\tau$ per loop iteration, whereas only one point multiplication involving the radix is usually required. The computational overhead of the additional point multiplication is minimal because a $\tau$-adic representation is involved. However, if a non-Koblitz curve is involved, the computation overhead of the additional point multiplication by the radix r is much larger. Therefore, the algorithms are not computationally feasible for non-Koblitz curve implementations.

#### 3.2.1 General DPA Countermeasure 1

By modifying Algorithm 3, it is possible to eliminate the second point multiplication operation. In the resulting algorithm, the point P is multiplied by the radix r each loop iteration, instead of the point Q. Similar to Algorithm 3 and Algorithm 4, the proposed SPA resistant algorithm and DPA countermeasure, presented as Algorithm 6, also uses a nonce j in an attempt to foil attackers. The nonce j is an (m+1)-bit sequence, which randomizes the elements of Q containing the results of k·P and k'·P. The nonce also randomizes the order in which calculations relating to k and k' are performed.

**Algorithm 6.** General DPA Countermeasure 1

Input: $k = \{k_{n-1}, k_{n-2}, \ldots, k_1, k_0\}_r$, P
Output: Q = k·P
1. Generate the nonce j, a random (m+1)-bit sequence
2. $kval[j_m] = k$, $kval[j_m'] = k'$
3. Q[1] = $\mathcal{O}$, Q[0] = $\mathcal{O}$
4. For i = 0 to m-1 do
   4.1. bit = $kval_i[j_i]$
   4.2. if (bit = 1) $Q[j_i] = Q[j_i] + P$
   4.3. bit = $kval_i[j_i']$
   4.4. if (bit = 1) $Q[j_i'] = Q[j_i'] + P$
   4.5. P = r·P
5. Return($Q[j_m]$)

By using corresponding bits from both k and k', one point addition operation is performed every loop iteration. The order that the bits of k and k' are involved in each loop iteration is dependent on a corresponding bit of the nonce j. Since j is a random nonce, it is suggested that an attacker is unable to determine the value of the bits of k. Furthermore, by using the nonce j to determine the order in which bits of k and k' are involved in each loop iteration and the index of k and k', the effectiveness of DPA attempts is reduced because each power trace depends on the nonce j. Therefore, successive power traces will be different.

Unfortunately, since the point P is multiplied by the radix r each loop iteration, windowing techniques cannot be efficiently employed. If the algorithm were modified to utilize a windowing technique, every point in the LUT would have to be multiplied by the radix r w-times per loop iteration, where w is the window width. Even for a Koblitz curve implementation, where the radix $\tau$ can be exploited, the computational overhead is substantial, making windowing impractical.

### 3.2.2 Random Array Element

The following DPA countermeasure is a variation of the random register renaming technique presented in [12]. Instead of a hardware implementation, the random register renaming technique is simulated in software with arrays (at the expense of memory). For a detailed description and analysis of the register naming technique, refer to [12].

By randomizing the array elements overwritten in each loop iteration, randomness is added to the power trace, because the power trace is dependent on the original and new data present in the array element. Therefore, attacking line 3.2 of the algorithm is less effective since the target address and overwritten data in each execution is changing randomly. The proposed DPA countermeasure is presented below as Algorithm 7. In the algorithm, $\oplus$ represents the exclusive-or logical operator.

**Algorithm 7.** Random Array Element Countermeasure

Input: $k = \{k_{m-1}, k_{m-2}, \ldots, k_1, k_0\}_r$, P
Output: Q
1. Generate the nonce j, a random (m+1)-bit sequence
2. $Q[j_m] = O$
3. For i = m-1 to 0 do
    3.1. $\delta = j_i \oplus j_{i-1}$
    3.2. $temp[\delta] = r \cdot Q[j_{i+1}]$
    3.3. $temp[\delta'] = P + temp[\delta]$
    3.4. $Q[j_i] = temp[\delta \oplus k_i]$
4. Return($Q[j_0]$)

## 4 Implementation and Performance Results

Each of the algorithms presented in the previous sections were implemented. A Koblitz curve over a binary finite field size of m=163 was used to implement each algorithm on the SC140. The performance of the algorithms, along with the performance of baseline point multiplication implementations was investigated. The cycle count, total operation counts and estimated memory overhead of the implementations is presented in Table 2. The resiliency of each implementation with respect to SPA and DPA attempts is stated, along with the elliptic curve applicable to each implemented algorithm.

By analyzing the fifth and sixth column of the table, the optimum algorithm for a specific implementation can be selected. For example, the memory overhead of the implementations is widespread. Therefore, according to the memory constraints of the target system, an optimum algorithm can be selected from the several options presented. The seventh, or last, column of the table can be used to compare the algorithms as well. However, the cycle counts listed do not directly correspond with the total operation count. This is because the polynomial inversion operation employed is not entirely SPA resistant. The computational cost of the operation is dependent on the input polynomial. Therefore, point addition operations do not operate in fixed time, resulting in slightly misleading cycle counts. For the implementation to be truly SPA resistant, a fixed time polynomial inversion operation is required.

**Table 2.** Algorithmic Implementation Performance and Memory Overhead

| Description | SPA/DPA Resiliency | Elliptic Curve | r-value (SE) | Total Operation Counts | Estimated Memory Overhead | Cycle Count |
|---|---|---|---|---|---|---|
| τ-adic Point Multiplication | None | General | N/A | 91·PA + 326·SQ | N/A | 2,676,000 |
| Algorithm 2, no windowing | DPA | Koblitz | 7 (5,040) | 97·PA + 614·SQ | 14·FF | 2,719,000 |
| Algorithm 2, using Algorithm 1 | Both | Koblitz | 5 (120) | 168·PA + 590·SQ | 10·FF | 4,661,000 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Algorithm 2, windowing (4-bits) | DPA | Koblitz | 6 (720) | 58·PA + 609·SQ | 44·FF | 1,614,000 |
| Algorithm 3 | Both | Koblitz | N/A | 163·PA + 652·SQ | 3·FF | 4,841,000 |
| Algorithm 4, windowing (4-bits) | Both | Koblitz | N/A | 93·PA + 662·SQ | 35·FF | 2,607,000 |
| Algorithm 5 | Both | Koblitz | N/A | 94·PA + 662·SQ | 35·FF | 2,630,000* |
| Algorithm 6 | Both | General | N/A | 163·PA + 326·SQ | 3·FF | 4,760,000 |
| Algorithm 7 | Both | General | N/A | 163·PA + 326·SQ | 3·FF | 4,760,000* |

* - Estimated cycle count, the algorithm was not actually implemented

## 5 SPA Analysis on the SC140

To determine the vulnerability of the implemented τ-adic point multiplication algorithms to SPA attempts, several power traces of an SC140 were recorded and analyzed. The goal was to determine k involved in the point multiplication operation. Again, a Koblitz curve over a binary finite field size of m=163 was used. Curve parameters can be found in [19]. In the analysis of each power trace, the first point addition operation is ignored because it is an easy case of the operation involving $\mathcal{O}$. The point addition is actually a point copy, which is computationally inexpensive and difficult to identify within a power trace. In §5 and §6, the y-axis of the figures is related to the input current of the processor. To approximate the actual current (in amps) of the SC140, divide the y-axis scale by five and add 74mA (due to DC coupling). Small k-values were analyzed to limit the power trace length because the equipment used to record the power traces is not well suited for long power traces.

First, power traces of the τ-adic point multiplication algorithm with and without windowing techniques must be analyzed to determine if SPA attacks are relevant on the SC140. The two power traces are listed as Figure 1 and Figure 2. Pertinent elliptic curve and finite field operations are labeled in each of the figures.

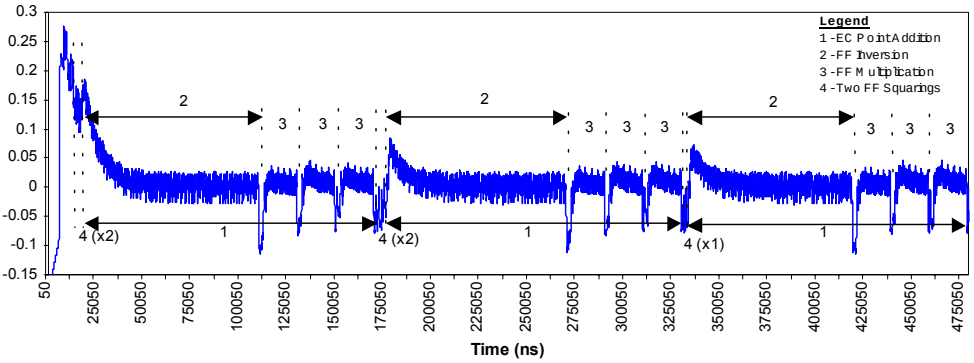


**Figure 1.** τ-adic Point Multiplication Power Trace (k=0x2B)

Figure 1 illustrates the power trace obtained when executing a τ-adic point multiplication on the SC140, with k=0x2B. The general case of elliptic curve point addition and finite field squaring operations are easily identified within the power trace. In the general case, where a point addition operation involves two unequal points, neither of which are the point at infinity or negatives of the other, the operation primarily consists of a finite field element inversion, followed by three consecutive multiplications. A multiple of two finite field squaring operations exist between each point addition operation when performing a τ-adic point multiplication. In Figure 1, more squaring operations exist between two point additions if the corresponding bit of k is zero. For example, the first two sets of consecutive finite field squaring operations require twice the execution time as the last, and because the implemented finite field squaring operation executes in fixed-time, the finite field squaring operation must be executed four times in the first two sets, and twice in the second set. Since the pattern of point addition and squaring operations can be determined

from the power trace, the general implementation of the τ-adic point multiplication operation is vulnerable to SPA attempts.

Figure 2 illustrates the power trace obtained when executing τ-adic point multiplication employing a windowing technique on the SC140. A window width of 4-bits and k-value of 0x53E4 were used to obtain the power trace. Since a window width of 4-bits is used, each set of consecutive finite field squaring operations consists of a multiple of eight operation calls. In this case, the point multiplication operation appears to be SPA resistant because each set of eight finite field squaring operations is followed by a point addition operation. However, hexadecimal zero coefficients can be determined from the power trace if they exist. In the case of zero hexadecimal coefficients, the corresponding point addition operation reduces to a point copy operation. Therefore, it is possible for an attacker to determine some of the bits of k.

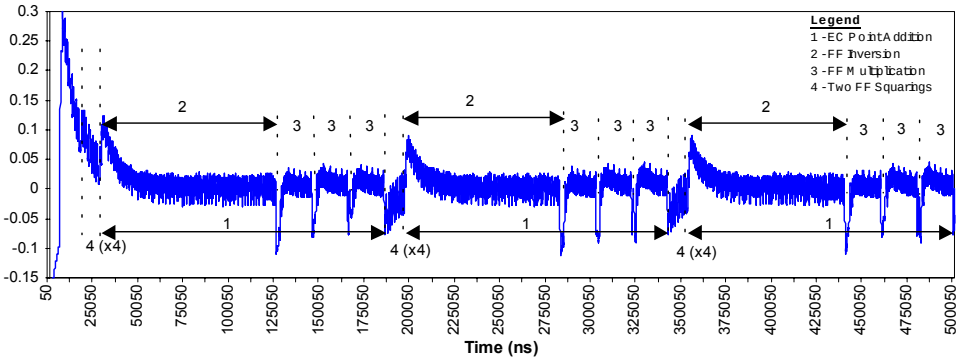


**Figure 2.** τ-adic Point Multiplication with Windowing Power Trace (k=0x53E4)

Figure 1 and Figure 2 show that by examining the power traces of the implementation of the τ-adic point multiplication technique with or without windowing, the value of the large integer involved in the operation can be partially or entirely determined. Therefore, an SPA resistant algorithm must be employed.

Algorithm 1 was investigated in an attempt to resist SPA attacks. Figure 3 illustrates the power trace of the implementation of Algorithm 1, with k=0xA. In the figure, the pattern of point addition and finite field squaring operations is fixed. Each pair of consecutive squaring operation calls is followed by a general case of the point addition operation. Therefore, the implementation of the point multiplication operation is not vulnerable to SPA.

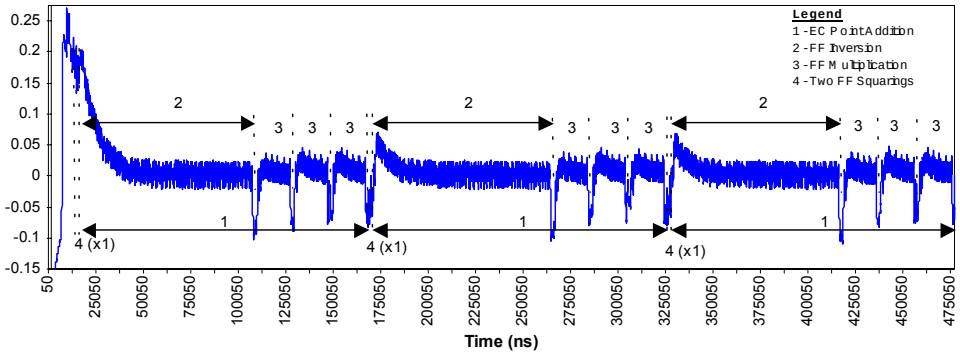


**Figure 3.** SPA Resistant Point Multiplication (Algorithm 1) Power Trace (k=0xA)

To ensure the rest of the algorithms are SPA resistant, power traces of each were analyzed. Figure 4 illustrates the power trace of the implementation of Algorithm 3, with k=0xB. Each of the other implementations of the algorithms presented in the paper resulted in similar power traces. The pattern of

point addition and finite field squaring operations is fixed in Figure 4. Therefore, the algorithm is SPA resistant.

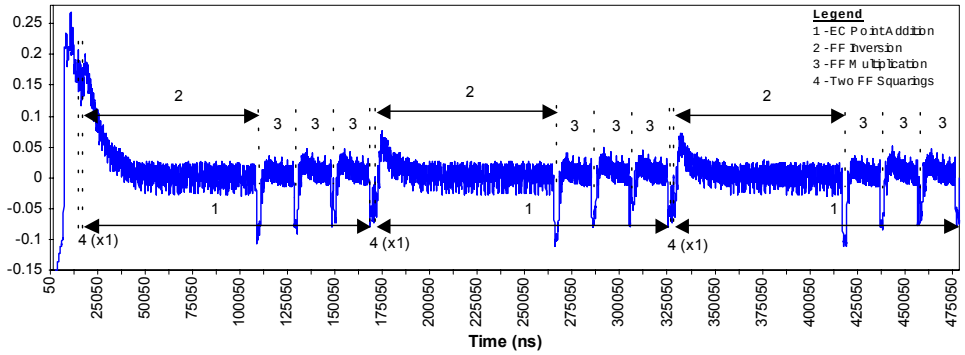


**Figure 4.** Koblitz Curve Specific DPA Countermeasure 2 (Algorithm 3) Power Trace (k=0xB)

## 6 Simulated DPA on the SC140

A DPA attack on the SC140 was simulated to determine if the processor is susceptible to such attacks. Sixty-four power traces, each involving two memory move instructions were recorded and analyzed. For the simulation, the SC140 was operating at 100 MHz, and a sample was recorded every 5ns. The y-axis in the figures is described in the previous section. A clear power trace that is only influenced by move instructions was obtained by surrounding the instruction by several NOPs. The data involved in each of the memory move instructions and their physical addresses in memory were exploited in the DPA attempts. First, a DPA attack based on data involved in the move instruction was attempted, which exploits the data correlation present in the power traces. Second, a DPA attack based on the memory address of the data involved in the move instruction was conducted. This type of attack attempts to exploit the address correlation in the power traces.

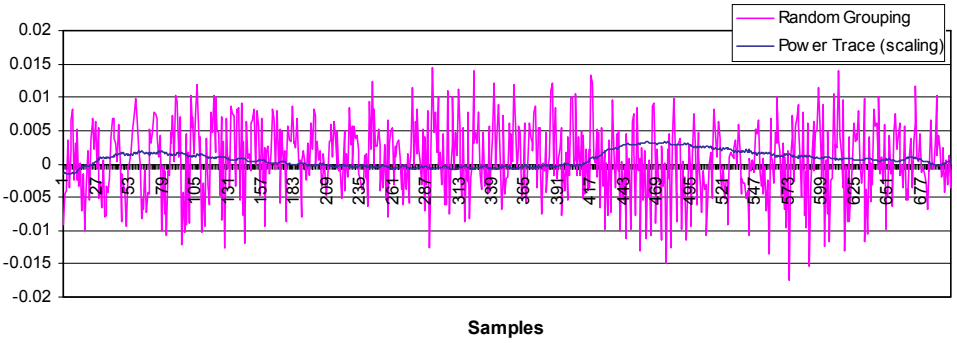


**Figure 5.** Differential Power Analysis – Random Grouping

Before the DPA attempts were performed, the power traces were grouped randomly, resulting in a baseline for comparison purposes. The power traces were partitioned such that no patterns existed in the bits of the address of the data and the actual data involved in the move instructions. The results are assumed to contain no correlation and are used to weigh the attacks, which attempt to exploit correlations within the power traces. Figure 5 presents the results of the DPA using a random grouping. A scaled version of the power trace with the two move instructions is also provided in the figure. The two bumps in the power trace represent the two memory moves. The bumps are much longer than a single clock cycle,

demonstrating the fact that both the current and several previous instructions influence the instantaneous power consumption of the SC140.

The first two DPA attempts, based on the data involved in the memory moves, were unsuccessful. To perform the attacks, the power traces were partitioned into two groups, based on the least significant bit of the data involved in each of the move instructions. The results obtained by analyzing the first move instruction are presented in Figure 6. The results do not differ from the random grouping results enough to state a correlation exists. Therefore, the DPA attacks based on the data involved in the move instruction were unsuccessful.

It is assumed that there is some correlation between the power traces and the data involved in the move instruction. Furthermore, it is believed that some other factor, such as the address of the data, reduced or masked the expected correlation. In either case, the correlation is weak enough that it is not easily identifiable. A more likely possibility is that the simulation was poorly organized, such that the data correlation is reduced, eliminated, or masked by other correlation factors.

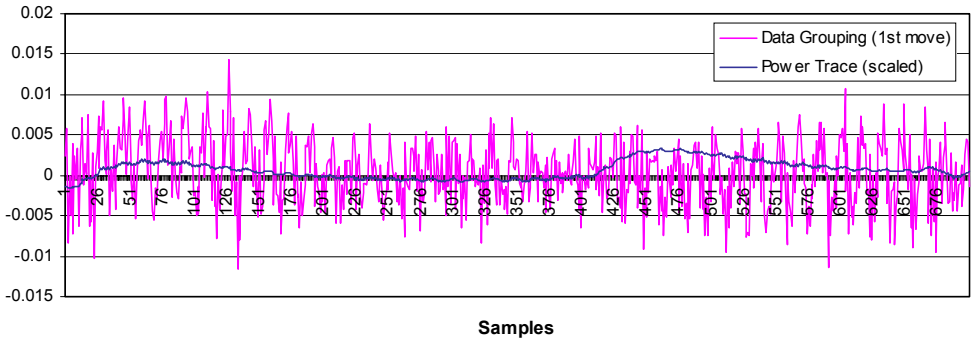


**Figure 6.** Differential Power Analysis - Data Grouping

The last DPA attempt, which was the most successful, was performed based on the address of the data. The power traces were partitioned into two groups based on the least significant bit of the address involved in the move instruction. Analyzing these data sets revealed a definite correlation. The analysis consisted of comparing the results from the random DPA grouping, the data DPA grouping, and the address DPA grouping. The results of the three groupings are presented graphically in Figure 7. When exploiting a DPA grouping based on address information, the results follow the second move in the power trace, demonstrating a definite correlation.

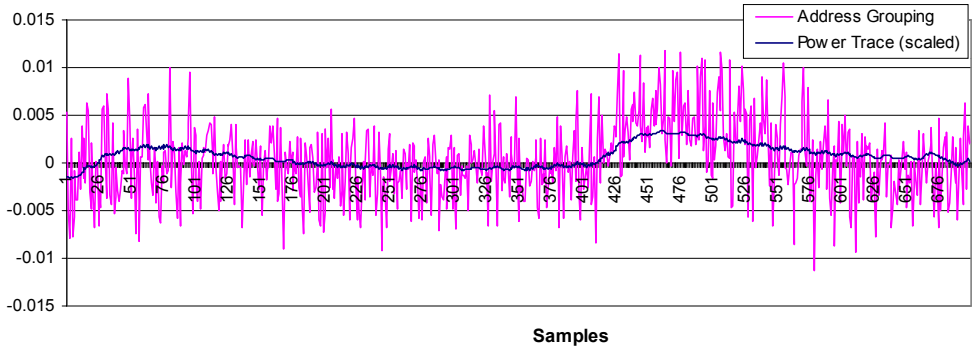


**Figure 7.** Differential Power Analysis – Address Grouping

## 7 Conclusion

In the paper, several DPA countermeasures are described and briefly analyzed. Countermeasures targeting both Koblitz and non-Koblitz curves are presented. Each countermeasure requires both the generation of

random data and a nonce in the form of a random sequence of bits to reduce the effectiveness of DPA attacks. It is believed that by utilizing random data, which defines the sequence of executions in the elliptic curve point multiplication operation, randomness is added to the power trace, reducing the effectiveness of DPA attacks.

The performance and memory overhead of each countermeasure is presented. In general, at the expense of memory overhead, a significant reduction in computational cost is obtained. By viewing the overhead of each countermeasure, the optimal algorithm for a specific implementation can be selected. Furthermore, it is possible to combine two (or more) of the proposed countermeasures presented in the paper to increase the effectiveness of resisting DPA attacks.

In the paper, DPA countermeasures were described at a high level of abstraction. More research is required to analyze the algorithms presented. To conclusively determine the effectiveness of the countermeasures presented, further analysis and possibly DPA attempts on actual implementations is required.

Actual power traces of implemented point multiplication operations are provided and examined. The power traces show the requirement of SPA countermeasures on the SC140 and the SPA resiliency of the proposed DPA countermeasures.

The natural ability of the SC140 to avoid DPA attacks was also investigated. Several DPA attempts were simulated on the DSP. DPA groupings based on 7-bit address information led to definite correlations, whereas the results of groupings based on the 6-bit data involved in the instructions were inconclusive.

For the first time, SPA countermeasures for Koblitz curves are presented using real power measurements. Furthermore, new SPA and DPA countermeasures are presented for both Koblitz and non-Koblitz curve EC implementations. Analysis of the countermeasures indicates that significant performance can be traded off against memory overheads.

## References


1. S. Chari, C. S. Jutla, J. R. Rao and P. Rohatgi, “Towards sound approaches to counteract power-analysis attacks”, in *Advances in Cryptology - CRYPTO '99*, LNCS 1666, p.398-412, Springer-Verlag, 1999.
2. C. Clavier and M. Joye, “Universal exponentiation algorithm: a first step towards provable SPA-resistance”, in *Workshop on Cryptographic Hardware and Embedded Systems- CHES 2001*, LNCS 2162, pp. 300-308, Springer-Verlag, 2001.
3. J.-S. Coron, “Resistance against differential power analysis for elliptic curve cryptosystems”, in *Workshop on Cryptographic Hardware and Embedded Systems*, LNCS 1717, pp.292-302, Springer-Verlag, 1999.
4. J.-S. Coron, M. Joye and C Tymen, “Differential analysis: attacks and countermeasures for elliptic curve cryptography”, 2002.
5. C. H. Gebotys and R. J. Gebotys, “Secure elliptic curve implementations: an analysis of resistance to power-attacks in a DSP processor”, in *Workshop on Cryptographic Hardware and Embedded Systems-CHES 2002*, 2002.
6. M. Hasan, “Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems”, in *Cryptographic Hardware and Embedded Systems - CHES 2000*, LNCS 1965, pp. 93-108, Springer-Verlag, 2000.
7. E. Hess, N. Janssen, B. Meyer and T. Schütze, “Information leakage attacks against smart card implementations of cryptographic algorithms and countermeasures”, Available at http://infilsec.com/papers/dpa/
8. K. Itoh, J. Yajima, M. Takenaka, and N. Torii, “DPA countermeasures by improving the window method”, in *Cryptographic Hardware and Embedded Systems – CHES 2002*, 2002.
9. M. Joye, C. Tymon, “Protections against differential analysis for elliptic curve cryptography”, in *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, pp. 377-390, Springer-Verlag, 2001.
10. P. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”, in *Advances in Cryptology - CRYPTO’96*, LNCS, pp. 104-113, Springer-Verlag, 1996.

11. P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis", in *Advances in Cryptology-CRYPTO'99*, LNCS, pp. 388-397, Springer-Verlag, 1999.
12. D. May, H. L. Muller and N. P. Smart, "Random register renaming to foil DPA", in *Workshop on Cryptographic Hardware and Embedded Systems- CHES 2001*, LNCS 2162, pp.2-38, Springer-Verlag, 2001.
13. Metrowerks Corporation, *CodeWarrior IDE*, version 4.1, build 696, 2000.
14. Metrowerks Corporation, *StarCore C Compiler*, vMtwk.Production 1.1, build 050901-1936, 2000.
15. B. Möller, "Securing elliptic curve point multiplication against side-channel attacks", in *Information Security – 4th International Conference, ISC 2001*, LNCS 2200, pp.324-334, Springer-Verlag, 2001.
16. Motorola, *SC140 DSP Core Reference Manual*. Motorola and Lucent Technologies Inc., Rev. 1, (2000). Available at http://www.motorola.com
17. K. Okeya and K. Sakurai, "On insecurity of the side channel attack countermeasure using addition-subtraction chains under distinguishability between addition and doubling", LNCS 2384, pp. 420-435, Springer-Verlag, 2002.
18. K. Okeya and K. Sakurai, "Power analysis breaks elliptic curve cryptosystems even secure against the timing attacks", in *Progress in Cryptology - Indocrypt 2000*, LNCS 1977, pp. 178-190, Springer-Verlag, 2000.
19. J. Solinas, "Efficient arithmetic on Koblitz curves", *Designs, Codes and Cryptography,* 19, pp.195-249, 2000.