Improved Error-detection and Fault-tolerance in ECSM Using Input Randomization

Agustin Dominguez-Oviedo and M. Anwar Hasan
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada, N2L 3G1

Abstract

Verification of the correctness of computations is very important to provide resistance against fault-based attacks. In elliptic curve cryptography (ECC), point validation (PV) alone is not sufficient against all fault analysis attacks. In this report error-detecting and fault-tolerant schemes for elliptic curve scalar multiplication (ECSM) are considered. For the purpose of detecting errors, we present structures based on re-computation and parallel computation along with PV. These structures use encoding techniques that rely on proprieties of elliptic curves and provide a high probability of detection of errors caused by faults occurred naturally or injected by an attacker. Additionally, we show that using parallel computation along with either PV or re-computation, it is possible to have fault-tolerant structures for the ECSM. Prototypes of the proposed structures for error detection and fault tolerance have been implemented and experimental results have been presented.

Keywords: Elliptic curve cryptography, fault-based attacks, error-detection, and fault-tolerance.

1 Introduction

Various researchers have emphasized the significance of cryptographic applications being resistant to side-channel analysis (e.g., reconstruction of a secret key from analysis of timing [17], power consumption signals [16], and electromagnetic emanations [1] during cryptographic operations). Another type of attacks that has received considerable attention is the fault analysis attack. First considered in [5] by Boneh, DeMillo, and Lipton, this attack consists in obtaining sensitive information from cryptosystems that are exposed to malfunctions during cryptographic computations. They described this attack on schemes such as the RSA and Rabin digital signatures. In order to avoid such attacks, they suggest verifying the correctness of computations in cryptographic applications.

In the past, a number of fault analysis attacks to elliptic curve cryptosystems (ECC) have been presented (see for example [3], [6]). One countermeasure to these attacks is to verify whether or not the points used in the computation are on the proper elliptic curve [2]. This countermeasure based on point validation (PV) alone is not sufficient against all fault analysis attacks on ECC. Recently in [4], Blömer, Otto, and Seifert presented a new ECC fault-based attack: Sign Change Fault (SCF) attack. This attack is for elliptic curves over prime fields where a sign change in a point implies only a change of sign of its y-coordinate. An interesting aspect of this attack is that the elliptic curve operations do not need to leave the original elliptic curve in contrast with those attacks described in [3], [6], and [2]. They assume that the attacker can induce a fault that produces a sign change into an intermediate point Q', such that $Q' \leftarrow -Q'$ during the elliptic curve scalar multiplication (ECSM) operation. After having a set of erroneous results due SCF attacks and the correct result Q, it is possible to recover the scalar k for an input pair (k, P). This type of attack is an example where PV is not a sufficient countermeasure and other approaches are needed for protection.

A straightforward countermeasure against an SCF attack is to use Montgomery's point multiplication algorithm [20], which does not use the y-coordinate for computing the ECSM. Another countermeasure presented in [4] uses a second elliptic curve whose order is a small prime number. This curve is utilized to define a named "combined curve" that is used to verify the final result to avoid this SCF attack. One disadvantage of this approach is that elliptic curve crypto-processors might have fixed parameters (e.g., those recommended by NIST). Hence, it might not be possible to redefine the curve parameters for having such a countermeasure.

In this report we extend our previous work [9] on error-detecting and fault-tolerant ECSM and present some structures that permit detection of errors in ECSM without modifying the curve parameters. These are based on re-computation and parallel computation along with PV. We use a number of encoding techniques that rely on properties of elliptic curves and provide a high probability of detection of errors caused by faults occurred naturally or injected deliberately by an attacker. In addition, we consider fault-tolerant ECSM. Error detection may be a sufficient countermeasure for many security applications, however fault-tolerant characteristic enables a system to perform its normal operation in spite of faults. For certain fault models, we propose structures that can perform correct ECSM operations in presence of faults that may occur in a limited number of ECSM modules, primarily due to natural causes such as, abnormal temperature, electromagnetic interference, or power supply changes. The encoding techniques used with the inputs can essentially prevent such structures from outputting incorrect results due to faults injected at precise locations in arbitrary number of ECSM modules by a sophisticated attacker.

The organization of the remainder of this report is as follows. In Section 2, we give a brief overview of ECC and some error detection strategies. Section 3 presents encoding schemes for error-detection for ECSM and probability

of undetected errors. In Section 4, fault-tolerant ECSM structures are given. Overhead costs and experimental results for the probability of undetected errors are presented in Sections 5 and 6 respectively. Finally, some concluding remarks are made in Section 7.

2 Background

2.1 Elliptic curve cryptography overview

An *elliptic curve* E over a finite field $GF(q)$ is defined by the following equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in GF(q). \tag{1}$$

For binary finite fields, from Equation (1), one can perform an admissible change of variables that transforms E to the non-supersingular curve:

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

Similarly for prime fields, E can be transformed to the curve:

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

The points (x, y) that satisfy Equations (2) or (3), along with the point at infinity ($\mathcal{O}$), and a particular operation known as point addition ($\uplus$), form an abelian group. The number of points on the elliptic curve E, denoted as $\#E$, is know as the *order* of E. If P is a point on the curve E and k is a positive integer, then the *elliptic curve scalar multiplication* (ECSM) is given as follows $kP = P \uplus P \uplus \cdots \uplus P$ (k times).

The elliptic curves obtained from Equations (2) and (3) with their respective point addition rules are for affine coordinates (i.e., (x, y)). If finite field inverse cost on $GF(q)$ is significantly more than finite field multiplication, then it may give a speed benefit to represent the points using a projective coordinates system (i.e., (X, Y, Z)). Among these projective systems, for the binary field $GF(2^m)$, the one by Lopez and Dahab [18] appears to be the most efficient, and is related to the affine system by $x = (X/Z)$ and $y = (Y/Z^2)$. For curves defined over the prime field $GF(p)$, an attractive projective representation is called the Jacobian system [24]. Its relation with the affine system is $x = (X/Z^2)$ and $y = (Y/Z^3)$.

2.2 Error detection strategies

In this subsection we give an overview of some techniques that can be used for error detection in ECSM. First, a method that is specific for applications using elliptic curves, namely PV process, is described. Next are given general re-computation and parallel computation schemes that can also be used for detecting errors in ECSM.

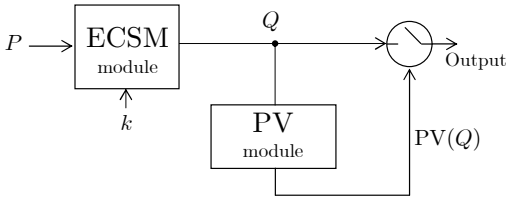

Figure 1: Point validation (PV) module after the ECSM module

2.2.1 Error detection using point validation (PV)

For ECC, PV can be used to detect errors during the ECSM. This is a basic countermeasure to prevent some fault attacks on ECC and it has been considered in [3], [6], [2], and [4]. A PV module is a computational module that takes a point Q as its input and checks whether the point is on the elliptic curve E (see Figure 1). If it do, only then an affirmative signal (say 'ok') is generated as output, i.e.,

$$\mathrm{PV}(Q) = \begin{cases} \mathrm{ok} = 1 & \text{if } Q \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The PV process requires only a few finite field operations and hence its implementation is relatively easy. If implemented in hardware as a separate module, the finite field multiplier in PV module could be designed to run at a much slower speed since PV is done just once for each scalar multiplication. On the other hand, the multiplier in the ECSM module is often optimized for speed since its operation is repeated many times.

Although, PV is an important countermeasure against some ECC fault based attacks, this process alone is not sufficient for some cases. For example, in the SCF attack the faulty points never leave the original elliptic curve E and the PV module alone cannot provide resistance against this attack. Accordingly, it is necessary to have added and/or other protections.

2.2.2 Error detection using re-computation and parallel computation

For applications where the time required for error detection is not critical, it is possible to compute the result twice (i.e., once at time t_0 and then again at time t_1) and then compare the two results to detect a possible error. This technique and its variants are used in various computing applications (e.g., integer multiplication); however to the best of our knowledge no previous article has reported an extension of these techniques to ECSM. Under this technique, a transient error occurring in any one of the two ECSM computations (i.e. at time t_0 or t_1), but not in both will be detected. However, if the computation module is permanently in a faulty state, the errors produced will remain undetected. Because of this, the re-computation at time t_1 is better done with a different method.

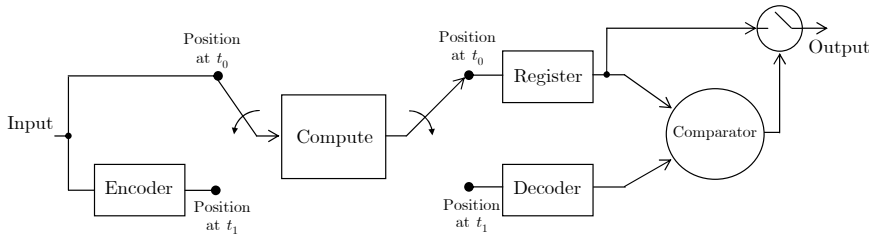

Figure 2: General re-computation based scheme (for ECC, *compute* module corresponds to ECSM module)

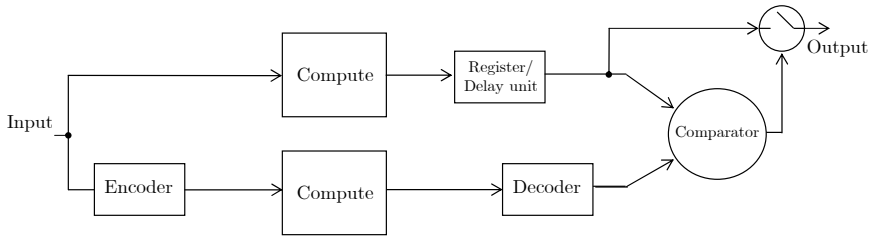

Figure 3: General parallel computation based scheme

A general re-computation based scheme that uses only one computing module is shown in Figure 2. The *compute* module is multiplexed between the upper and the lower data paths at time t_0 and t_1 respectively. The result of the upper data path is stored in a register. The encoder in the bottom data path of the figure produces a different set of inputs to the compute module and the decoder transforms the output of the module back to the original domain for final comparison. The encoding and the decoding processes enable us to detect errors caused by a permanent fault in the compute module. Encoding and decoding can be carried out through a number of approaches. For conventional number systems the reader is referred to [22], [21], and [14]. In Section 3, we provide encoding and decoding processes that are suitable for ECSM.

Similar to re-computation, where we have one module that performs the computation twice, we can have two independent modules working in parallel. A general parallel computation based scheme is given in Figure 3, where the register or the delay unit is to synchronize the inputs to the comparator.

3 Encoding/Decoding and Error-detection for ECSM

In this section, improved error-detecting structures for ECSM are proposed. Here, we focus on a high-level design, where the ECSM module is the main block implemented in hardware to accelerate some ECC applications and may become faulty either by natural causes or by deliberate attacks from an adversary. Other modules used in our error-detecting structures are much less complex than ECSM module and are assumed to be implemented in a secure environment – either in software or hardware.

Throughout the rest of this report, we use the following assumptions:

- The input point P is verified to be on the valid elliptic curve before each ECSM computation. This validation is especially important for preventing attacks as described in [3] and [2].
- The order of a selected point P is a sufficiently large prime. This check guarantees that P is not in a small subgroup of E of order dividing the cofactor h [13].
- An appropriate elliptic curve has been selected (e.g., using the guidelines of a recognized standard such as FIPS 186-2 [11]).

3.1 Encoding/decoding for ECSM

The encoding/decoding process in Figures 2 and 3 plays an important role in the detection of errors caused by faults in the compute (i.e., ECSM) module. Without the encoding/decoding the re-computation based scheme in Figure 2, the ECSM module – with permanent and/or transient faults injected by a sophisticated attacker that affect the module's two 'runs' in the same way – is likely to produce erroneous but the same results at times t_0 and t_1, and the error cannot be detected by the comparator. Similarly, for the parallel computation based scheme in Figure 3, if the two ECSM modules have the same permanent and/or transient faults, in the absence of the encoding/decoding the two modules are likely to generate erroneous but the same results. Again, such errors are not detected by the comparator.

Below we present encoding/decoding schemes suitable for ECSM operations. These schemes are primarily based on properties of elliptic curves. For a given input pair k and P, a 'good' encoding scheme will produce input representations that are different for the lower and the upper data paths (see Figures 2 and 3), and hence faults – whether or not identical on the two data paths – are likely to affect the two ECSM operations differently. Consequently, the two values to be compared by the comparator are also likely to be different.

3.1.1 Encoding for input point P

Taking advantage of the simplicity of negating an EC point, it is possible to use the negative of a point as the encoded input [9]. For a point P given in the affine coordinate system, it is well known that the point negation is simply

$$-P = -(x, y) = \begin{cases} (x, x + y) & x, y \in GF(2^m), \\ (x, -y) & x, y \in GF(p). \end{cases} \tag{4}$$

For projective coordinates, namely the Lopez and Dahab system for curve E over $GF(2^m)$ and the Jacobian system for E over $GF(p)$, the point negation is

$$-P = -(X, Y, Z) = \begin{cases} (X, XZ + Y, Z) & X, Y, Z \in GF(2^m), \\ (X, -Y, Z) & X, Y, Z \in GF(p). \end{cases} \tag{5}$$

In Figure 2, if the input is point P, then the encoder performs the point negation in accordance with Equation (4) or (5). For the encoded input, the output of the ECSM module is $k(-P) = -kP$. Hence the decoder in Figure 2 also performs a point negation to generate the expected output kP.

Although, the encoding (and decoding) using point negation is simple – for projective coordinates one multiplication and one addition for E over $GF(2^m)$, and only one addition for E over $GF(p)$ – it is important to note that such an encoding scheme changes only the y-coordinate of the input point, while the x- and, if applicable, the z-coordinate remain unchanged. As a result, even in the presence of some faults in the ECSM module, it is possible that the comparator in Figure 2 gets two equal but incorrect points at its input and generates an 'ok' signal.

To change all the coordinate values of the input point we can use a property of projective coordinate systems, which consists of having multiple representations for a given point. This principle is known as point randomization [7] and is applicable to all the projective coordinate systems [15]. For the Lopez and Dahab and the Jacobian projective systems, Equations (2) and (3) become

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4 \tag{6}$$

and

$$Y^2 = X^3 + aXZ^4 + bZ^6 \tag{7}$$

respectively. It is easy to verify that trios $(\gamma X, \gamma^2 Y, \gamma Z)$, where $\gamma \in GF^*(2^m)$ satisfies Equation (6) and has the affine representation as that of (X, Y, Z). Thus, these trios can give different projective representations of a single point on the curve defined by Equation (2). Similarly trios $(\gamma^2 X, \gamma^3 Y, \gamma Z)$, where $\gamma \in GF^*(p)$, have different projective representations of a single point in the curve defined by Equation (3).

Based on the above discussion, a single projective representation (X, Y, Z) can be encoded, in some random way, to one of the $q - 1$ possible projective representations $(\gamma^c X, \gamma^d Y, \gamma Z)$, where for the Lopez and Dahab system, $c = 1$, $d = 2$, and $\gamma \in GF^*(2^m)$; and for the Jacobian system $c = 2$, $d = 3$ and

$\gamma \in GF^*(p)$. Since (X, Y, Z) and $(\gamma^c X, \gamma^d Y, \gamma Z)$ correspond to the same point, the two ECSM operations, i.e., $k(X, Y, Z)$ and $k(\gamma^c X, \gamma^d Y, \gamma Z)$, result in the same point on the curve i.e.,

$$k(X, Y, Z) \sim k(\gamma^c X, \gamma^d Y, \gamma Z). \tag{8}$$

One implication of Equation (8) is that if the encoder in Figure 2 or 3 is only for the following mapping

$$(X, Y, Z) \mapsto (\gamma^c X, \gamma^d Y, \gamma Z), \tag{9}$$

then the decoder is not needed.

3.1.2 Encoding for scalar k

It is well known that the order of any point P divides the order of the group $\#E$, i.e., $(\#E)P = \mathcal{O}$. Let $k' = \#E - k$. Then

$$k'P = (\#E - k)P \equiv -kP.$$

Thus k can be simply encoded to k', which can be viewed as some kind of scalar negation operation, and the corresponding decoding process involves a point negation to convert $-kP$ to kP. However, this decoding process can be omitted if the correspondent input point P is also negated in the encoding process (i.e., $k'(-P) = kP$) [9].

For the scalar k, more complex encoding schemes are possible. When k is fixed (e.g., in ECDSA [11]), in order to resist differential power analysis attacks, it has been suggested [7] to randomize k by the following mapping

$$k \mapsto k'' = k + j(\#E), \tag{10}$$

where j is a random integer of at least 20 bits long. Such randomization can also be used as an encoding scheme, since

$$k''P = (k + j(\#E))P \equiv kP.$$

Such encoding requires an integer multiplication, however no computations are needed for decoding.

Among other possible encoding schemes, the binary unsigned representation of k can be converted into one of the many binary signed representations of k. This conversion/encoding can be done in a random way. For an m-bit integer k, there are $O(3^{\lfloor \frac{m}{2} \rfloor})$ different binary signed digit representations on average [10]. This however requires the ECSM module to support the double-and-add/sub algorithm and is not considered here.

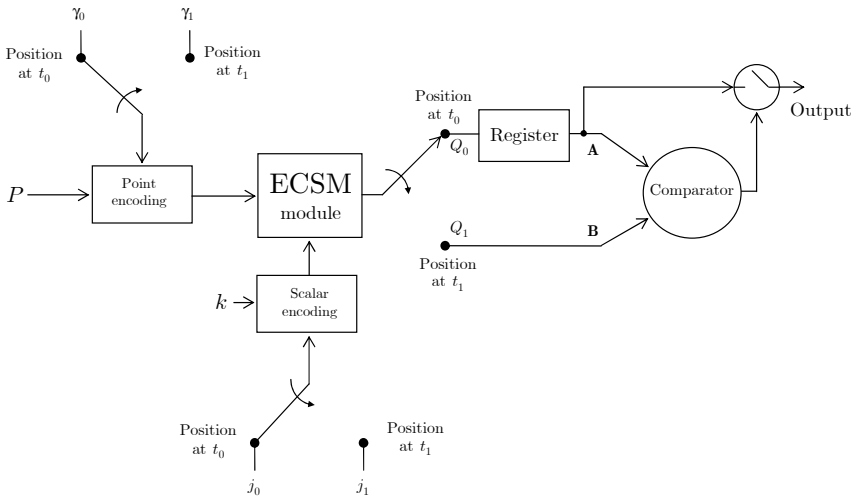

Figure 4: ECSM using full re-computation with point and scalar randomization (RC)

3.2 Error-detecting structures and probability of undetected error

As stated before, for the purpose of encoding one can map input pair k and $P = (X, Y, Z)$ to $k + j\#E$ and $(\gamma^c X, \gamma^d Y, \gamma Z)$ respectively. These encoding schemes produce different representations of inputs that are equivalent to the original input and do not require the decoding process. In addition, as shown in Section 6 with experimental results, these two encoding schemes lead to a lower probability of undetected errors when compared with other encoding schemes discussed earlier, namely $P \mapsto -P$ alone or combined with $k \mapsto k' = \#E - k$. As a result, in this work we use these two encoding schemes for error detection in ECSM. In particular, for re-computation based error detection, at t_0 we compute the ECSM with (k, j_0, P, γ_0) as inputs, and then at t_1 compute another ECSM with (k, j_1, P, γ_1), where j_0 and j_1 are two random integers of appropriate length (say 20 bits for a 160 bits long k) and γ_0 and γ_1 are two random non-zero elements of the underlying finite field. In the comparator unit, the outputs of the two ECSM operations are converted from projective to affine representations and then a bit-wise comparison is performed. If the two affine representations are the same, one of them is produced as the final output. Otherwise, no final output is given. This scheme is shown in Figure 4 and we refer to it as full re-computation based scheme or RC for short.

Because of random γ_i and j_i for $i = 0$ and 1, one can assume that in the presence of faults – whether identical or not – the ECSM module's output Q_i is a random trio (X_i, Y_i, Z_i) of finite field elements. If the fault makes the ECSM module to produce an incorrect result, then $Q_i \neq kP$. Since the number of

elements in the finite filed is q and Q_i has $q-1$ different projective representations, the probability that $Q_0 = Q_1$ and $Q_i \neq kP$ (i.e., undetected error) can be expressed as follows

$$\Pr(\text{undetected error})_{\text{RC}} \approx \frac{q-1}{q^3} \approx \frac{1}{q^2}. \tag{11}$$

For many of today's security applications , where $q \approx 2^{160}$, the above probability of undetected error is quite small. It can be further reduced by including PV to RC to check whether Q_0 or Q_1 are on the proper curve. For the verification of Q_i, we can write

$$\Pr(Q_i \in E) = \frac{(\#E-1)(q-1)+1}{\text{Number of finite field element trios}} \approx \frac{q^2}{q^3} = \frac{1}{q}. \tag{12}$$

For practical purposes it appears to be sufficient to verify only one point – preferably Q_0 while it is waiting in the register (see Figure 4) – if a final output is generated when the following two equations are simultaneously satisfied

$$Q_0 = Q_1 \text{ and}$$

$$\text{PV}(Q_0 \text{ or } Q_1) = \text{ok}.$$

The RC scheme along with PV as described above will be referred to as RC_PV. Based on the above discussion, its probability of undetected error is

$$\Pr(\text{undetected error})_{\text{RC\_PV}} \approx \frac{1}{q^2} \times \frac{1}{q} = \frac{1}{q^3}. \tag{13}$$

The counterpart of RC that uses parallel computation is shown in Figure 5. This parallel computation based scheme (or PC for short) can detect any error confined in one module. Additionally if both ECSM modules have errors, since they use different input representations, the probability of having equal erroneous outputs from these modules is low as given in Equation (11). To reduce the probability of undetected errors similar to RC_PV, we can use PV at location A or B (in Figure 5) and we refer the resulting scheme as PC_PV. The probability of undetected error for PC_PV is the same as given in (13).

3.3 Error detection using partial re-computation

The main penalty of using full re-computation (i.e., using RC or RC_PV) is that it doubles the running time of ECSM. In applications where the ECSM module is subject to only permanent faults injected by an attacker or caused naturally, the running time can be reduced by performing a partial re-computation (RC_partial) as discussed below.

Under RC_partial, input point P is encoded twice using point randomization described in (9) so that it has different representations at t_0 and t_1. Input scalar k is encoded to k'' using (10) for ECSM at t_0; but no encoding is done for t_1 and only a few bits (say, least significant l bits, where $0 \leq l \leq m$) of the

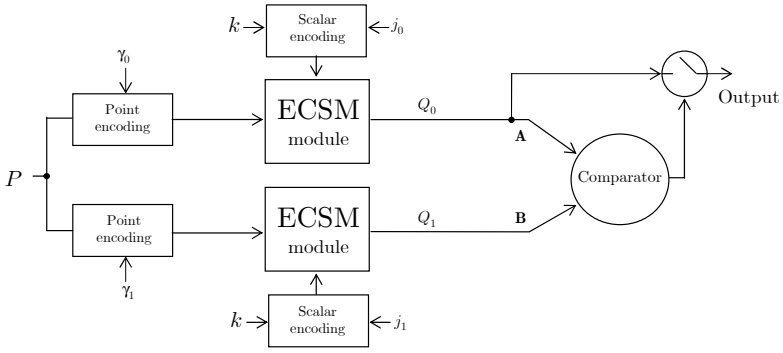

Figure 5: Parallel computation based ECSM with point and scalar randomization (PC)

encoded k are used. Let these l bits correspond to integer $k_s = \sum_{i=0}^{l-1} 2^i k_i''$, where $k_i'' \in \{0, 1\}$.

Based on the above discussion, at time t_0, an ECSM is computed with (k, j_0, P, γ_0). Let us denote the output of this computation as Q_0. As part of the computation, assuming a left-to-right double-and-add algorithm, $k_s P$ is expected to be generated after l iterations of the algorithm. This value (i.e., point $k_s P$) or an erroneous version of it – in case there are errors due to faults – is saved for a later comparison. Let us denote the saved value as $Q_{0,\text{ partial}}$.

At time t_1, an ECSM is computed with $(k_s, 1, P, \gamma_1)$. Since k_s is l bits long, this computation requires $l - 1$ point doublings and $H(k_s) - 1$ point additions, where $H(k_s)$ denotes the Hamming weight of k_s. Let us define the output of this ECSM as Q_1. If the affine representations of Q_1 and $Q_{0,\text{ partial}}$ are the same, then Q_0 computed at t_0 is produced as the final result.

The probability of undetected errors in the RC_partial scheme depends on the value of l. If l takes the maximum value i.e. $l = m$, then RC_partial is the same as RC discussed earlier and doubles the running time but the probability of undetected errors is very low as given in Equation (11). On the other hand if l is minimum i.e. 0, then there is no re-computation at all and no errors will be detected unless other techniques such as PV are used. For practical purposes, l may be taken as a small fraction, say 10%, of the number of bits in k and still achieve a low probability of undetected error since the point encoding technique described in (9) produces two random projective representations of the point at t_0 and t_1.

4 Fault-tolerant structures for ECSM

In this section, methods for fault tolerance for ECSM are presented. First, a classical example of hardware redundancy, triple modular redundancy (TMR), is considered for ECSM. Then by taking advantage of the simplicity of PV

operation, we present a double modular redundancy (DMR) based fault-tolerant scheme. Finally, by combining parallel computation with re-computation, we present another ECSM fault-tolerant structure that is as robust as TMR, and achieves an area efficiency close to DMR. For each of the fault-tolerant schemes, we also give its reliability, which is defined as its characteristic expressed by the probability that it will perform its function [23]. Finally, we provide a reliability comparison among these schemes.

4.1 TMR based fault-tolerant ECSM

Traditional TMR utilizes three elements performing the same operation. In the context of the work presented here, these elements would correspond to ECSM modules as shown in Figure 6. Suppose that we have this TMR based ECSM scheme without point and scalar randomization (i.e., $\gamma_i = 1$ and $j_i = 0$ for $i \in \{0, 1, 2\}$). In such a case, as long as two or all three ECSM modules yield correct results, the majority voter produces a correct final output. When two or more ECSM modules become faulty by natural causes or by some simple malicious action by a less sophisticated attacker, then the faulty modules are likely to produce different but incorrect results, and the majority voter will produce no final output. However, when a sophisticated attacker can deliberately inject faults that may cause two or more ECSM modules to generate the same but incorrect result, then the majority voter will produce an erroneous output and the TMR scheme will fail.

In an attempt to reduce the chance of TMR producing an erroneous result, we proceed as follows. Scalar k and point P, which are inputs to the ECSM modules, are encoded using the randomization techniques discussed in Section III. The ECSM outputs are connected to a secure majority voter implemented in either software or hardware. The majority voter transfers the projective coordinate output of each of the ECSM module to the affine coordinate, and produces a final result Q_{TMR}, which is the majority of the three affine coordinate representations. If there is no majority, then no final output is generated. Clearly, if errors occur in only one of the ECSM modules then their effects can be masked and a correct final output can be obtained.

Assume that an attacker can inject the same fault, transient or permanent, in two or all three modules in an attempt to make the faulty modules produce the same but incorrect result. In such circumstances, the point and scalar encodings help to keep the risk of giving an incorrect result to a low level. This is explained as follows. Because of the encoding of inputs, in the presence of faults each ECSM module is expected to produce a random projective representation, which after being transformed to the affine coordinates will correspond to one of the q^2 possible affine representations. An erroneous final result is produced by the majority voter if two or all three affine representations are the same but

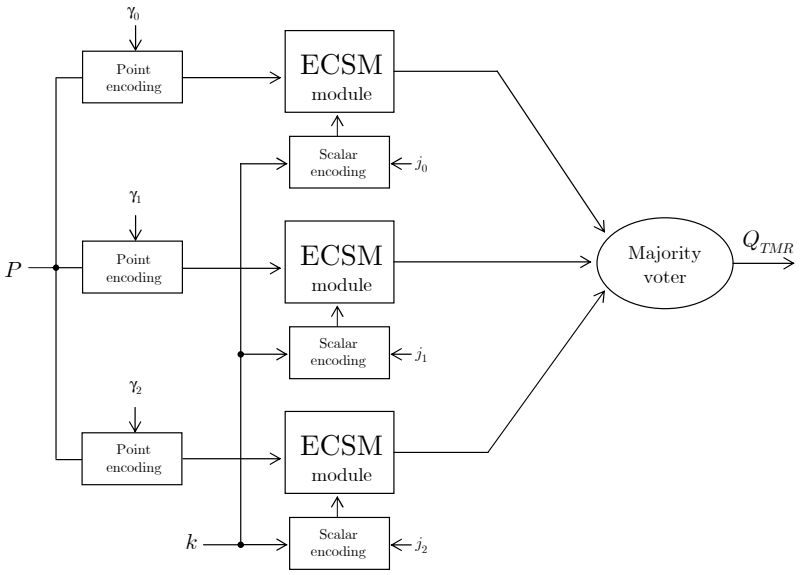

Figure 6: TMR based ECSM

incorrect. Thus the probability that the final result has an error is

$$\begin{aligned}\Pr(Q_{TMR} \neq kP) &\approx \binom{3}{2} \times \frac{q^2(q^2-1)}{q^2 \times q^2 \times q^2} + \frac{q^2}{q^2 \times q^2 \times q^2} \\ &= \frac{3}{q^2} + \frac{1}{q^4} \approx \frac{3}{q^2} \quad \text{(for large } q\text{)}. \end{aligned} \tag{14}$$

As an improvement, if PV is used for verifying whether Q_{TMR} is on E or not, then the probability of an erroneous final result can be reduced to about $\frac{3}{q^3}$ since

$$Pr(Q_{TMR} \neq kP \text{ and } Q_{TMR} \in E) \approx \frac{1}{q} \times \Pr(Q_{TMR} \neq kP).$$

We note that TMR can be viewed as a special case of an N-modular redundant system with N modules where faults confined in up to $\lfloor \frac{N-1}{2} \rfloor$ modules can be masked.

4.2 DMR based fault-tolerant ECSM

In applications where it is important to reduce silicon area, perhaps at the expense of increases in the probability of incorrect result, instead of using the TMR based scheme discussed earlier, one can use the dual modular redundant (DMR) system combined with PV as shown in Figure 7. As it can be clearly seen, compared to TMR, DMR uses one less ECSM module. As shown in Figure 7, the two ECSM operate in parallel and their outputs are verified by

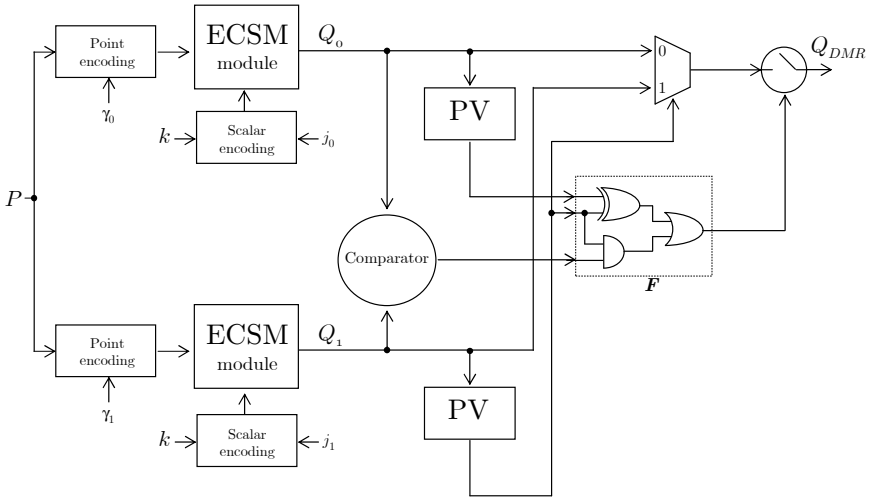

Figure 7: DMR based fault-tolerant ECSM

PV modules. The F block is used to stop the system output if $Q_0, Q_1 \notin E$; or when $Q_0 \neq Q_1$, and $Q_0, Q_1 \in E$. The final output of the DMR based system is given as follows

$$Q_{DMR} = \begin{cases} Q_1 \text{ (or } Q_0) & \text{if } \texttt{PV}(Q_0 \text{ or } Q_1) = \text{ok and } Q_0 = Q_1 \\ Q_i & \text{if } Q_i \in E \text{ and } Q_{\bar{i}} \notin E \\ \text{no output} & \text{otherwise.} \end{cases}$$

This DMR based scheme can clearly tolerate any faults confined in only one of the two modules and can produce the correct final output. Additionally, the scheme can detect some situations where errors exist in the outputs of both modules and hence helps to avoid producing erroneous results at the final output. However, there are two cases of reasonably low probability when this scheme fails and gives an incorrect result: firstly, if $Q_0 = Q_1$ and $Q_0, Q_1 \in E \setminus kP$; secondly, when $Q_0 \neq Q_1$, $Q_i \in E \setminus kP$, and $Q_{\bar{i}} \notin E$ for $i = 0$ or 1. Then, the probability of giving an incorrect result when both ECSM modules are in error is

$$\Pr(Q_{DMR} \neq kP) \approx \frac{1}{q^3} + 2\left(1 - \frac{1}{q^2}\right)\left(\frac{1}{q}\right)\left(1 - \frac{1}{q}\right) \approx \frac{2}{q} \quad \text{(for large } q\text{)}. \quad (15)$$

4.3 Parallel and re-computation based fault-tolerant ECSM

The above DMR based fault-tolerant scheme reduces the silicon area requirement but increases the probability of incorrect result. It is possible to have a fault-tolerant ECSM system that is as area efficient as DMR and has the same low probability of incorrect result as TMR. To this end, one can use a parallel and re-computation (PRC) based scheme shown in Figure 8. In

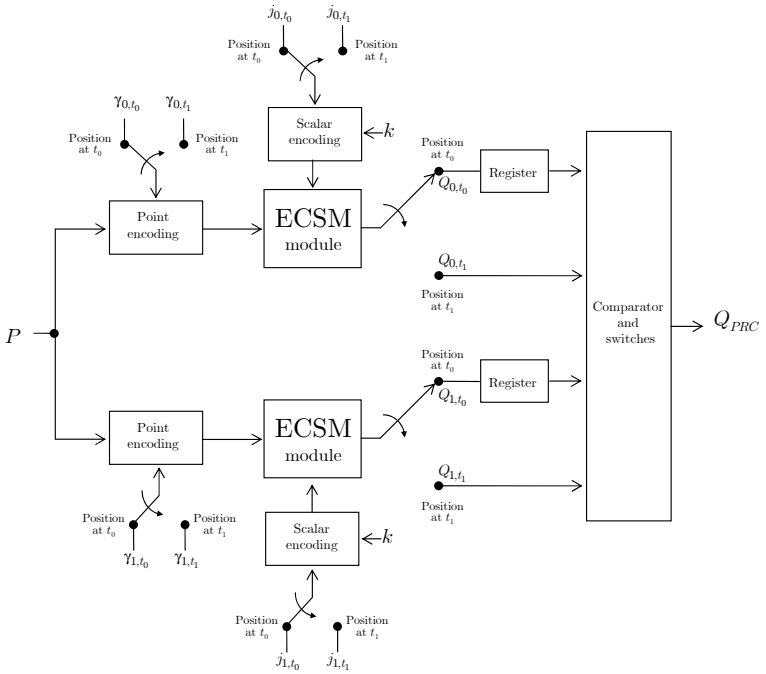

Figure 8: Parallel and re-computation (PRC) based fault-tolerant ECSM

PRC, at time t_0, with input $(k, j_{0,t_0}, P, \gamma_{0,t_0})$ and $(k, j_{1,t_0}, P, \gamma_{1,t_0})$ the two ECSM modules produce Q_{0,t_0} and Q_{1,t_0} respectively. If the affine representations, i.e., $\texttt{aff}(Q_{0,t_0})$ and $\texttt{aff}(Q_{1,t_0})$ are the same, then the final output $Q_{PRC} = \texttt{aff}(Q_{0,t_0}) = \texttt{aff}(Q_{1,t_0})$. Otherwise the ECSM modules perform re-computations with $(k, j_{0,t_1}, P, \gamma_{0,t_1})$ and $(k, j_{1,t_1}, P, \gamma_{1,t_1})$ and produce Q_{0,t_1} and Q_{1,t_1}. If errors are confined in only one of the ECSM modules, then for only one of the two values of i, i.e., either $i = 0$ or $i = 1$, the following is true

$$\texttt{aff}(Q_{i,t_0}) = \texttt{aff}(Q_{i,t_1})$$

which is also the final output Q_{PRC}.

An erroneous final output Q_{PRC} may be produced if both the ECSM modules produce the same but incorrect result. For the PRC operation described above, the probability that an erroneous final result is produced is

$$\Pr(Q_{PRC} \neq kP) \approx \frac{3}{q^2} \tag{16}$$

which matches (14) for TMR. Like TMR, PRC can be equipped with PV at the output so that the probability of an erroneous result can be further reduced by a factor of q.

We note that the re-computation in the PRC based scheme is performed when one or both ECSM modules generate incorrect results and hence doubles

the running time under such situations. If both ECSM modules are properly working, then the running time of PRC is as the same as TMR and DMR.

4.4 Effect on reliability

Note that compared to the ECSM module, each of the other modules in Figures 6, 7, and 8 namely the majority voter, the point and the scalar randomization modules, PV modules, comparators, registers, multiplexors would require a lot less hardware in practice. Thus, these small components can be implemented such that each of those will have a reliability factor that is close to unity. Then, for the case of TMR based ECSM the reliability is

$$R_{TMR} \approx r_{ECSM}^3 + 3r_{ECSM}^2(1 - r_{ECSM}) = 3r_{ECSM}^2 - 2r_{ECSM}^3, \qquad (17)$$

where r_{ECSM} is the reliability factor of a single ECSM module. On the other hand, for both DMR and PRC based ECSM schemes their reliability $R_{DMR/PRC}$ is related to the probability that at least one of the two ECSM modules work without errors. The resulting expression for $R_{DMR/PRC}$ is:

$$R_{DMR/PRC} \approx r_{ECSM}^2 + 2r_{ECSM}(1 - r_{ECSM}) = 2r_{ECSM} - r_{ECSM}^2. \qquad (18)$$

If we subtract Equation (17) from (18) we obtain

$$\begin{aligned} & R_{DMR/PRC} - R_{TMR} \\ = \; & 2r_{ECSM} - r_{ECSM}^2 - (3r_{ECSM}^2 - 2r_{ECSM}^3) \\ = \; & 2r_{ECSM}(1 - r_{ECSM})^2. \end{aligned}$$

Since $0 < r_{ECSM} < 1$, $R_{DMR/PRC} - R_{TMR}$ is always positive, i.e., we can state that the reliabilities of DMR and PRC based schemes are greater than that of the TMR based system (see Figure 9). The improved reliability of DMR and PRC schemes is primarily due to its fewer number of ECSM modules compared to the TMR.

5 Overhead cost

Similar to [9], we have used the performance results given in [19] for ECSM operation, where a NIST recommended elliptic curve over $GF(2^{163})$ has been used. Then using the same FPGA (i.e., Xilinx Virtex 2000E), we have implemented the rest of the components (i.e., PV module, point randomization module, scalar randomization module, register, comparator, majority voter, and multiplexor) required in the error-detecting and fault-tolerant ECSM structures presented in Sections 3 and 4. We also note that these components must be implemented in a secure environment so that they are not vulnerable against faults.

For some modules namely, PV, point randomization, comparator, and majority voter, in order to optimize the area requirement we have used a low speed

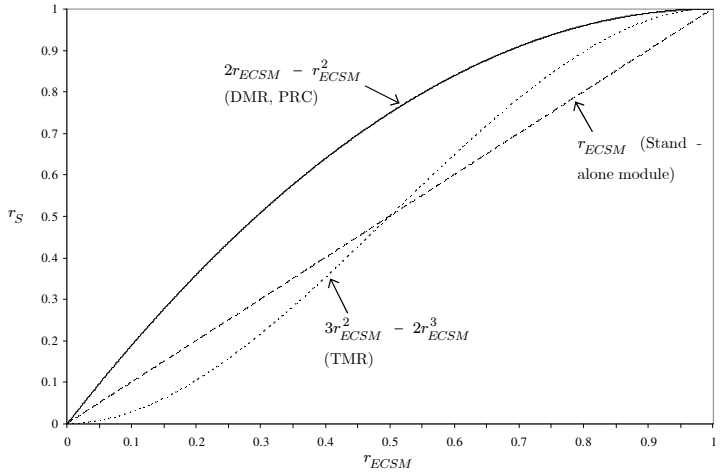

Figure 9: Reliability comparison between TMR and PRC schemes

multiplier which occupies 286 slices and 163 cycles for each finite field multiplication. This values contrast with the 2364 slices and 4 cycles required for each finite field multiplication in the ECSM module. The results of performance and cost for these extra modules are given in Table 1.

Element	**Slices**	**Cycles**
PV module	760	824
Point randomization module	528	818
Scalar randomization module[1]	815	18
Register	245	1
Comparator	714	2013
Majority voter	989	2013
Multiplexor	245	0

[1] size of $j = 20$ bits

Table 1: Cost and performance for the extra modules used for error-detecting and fault-tolerant structures

Incorporating the extra modules with the ECSM modules as shown in the error-detecting structures of Section 3, we obtain the time and space complexity results given in Table 2. The corresponding results for the fault-tolerant structures of Section 4 are shown in Table 3.

The clock cycle data in Tables 2 and 3 correspond to the time needed to produce a final ECSM output given input pair k and P. In applications where many such ECSM operations are to be performed, a more important measure of performance is the throughput, which can be defined as the number of ECSM outputs per unit of time. In the rightmost column of Table 2, we give ratios

ECSM with	Figure No.	Slices	Cycles	Throughput ratios
no error detection	NA	5009	17424	1
PV	1	5769	18248	1
RC	4	7311	37152	$\frac{1}{2}$
RC_PV	4[1]	8071	37152	$\frac{1}{2}$
RC_partial[2]	NA	7311	21676	$\frac{1}{1+l/m}$
PC	5	12075	20809	1
PC_PV	5[1]	12835	20809	1

[1] attaching one PV module at either location A or B [2] size of $l = 16$ bits

Table 2: Performance and cost for error-checking ECSM systems over $GF(2^{163})$

of the throughput with no error detection to that with various error detection schemes, such as PC, RC, RC_PV, etc. Similarly, in Table 3 we give such ratios for fault-tolerant schemes. In determining the ratios, we assume that the operations of an ECSM module and comparator/PV can be overlapped. The ratio results show a number of trade-offs one can consider between area and throughputs.

Scheme	Figure No.	Slices	Cycles	Throughput ratios
TMR based fault-tolerant ECSM	6	17359	21627	1
DMR based ECSM	7	13081	20809	1
PRC based ECSM	8	13301	20809[1]	1[1] or $\frac{1}{2}$

[1] if no error is detected at t_0, otherwise the running time is doubled/throughput is halved

Table 3: Performance and cost for fault-tolerant ECSM systems over $GF(2^{163})$

For the FPGA implementation described here, the clock frequency is 66 MHz. At this speed, the ECSM with the PV module, as shown in the first row of Table 2, will require $18248/(66 \times 10^6) \approx 276.48$ μsec. On the other hand, if all modules are implemented in software, then using the results reported in [12] for finite field arithmetic units, an ECSM unit with the PV module will require approximately 5.38 msec. These results may vary depending on the target platform and the level of optimization used; nevertheless, they show the speed advantage of using hardware over software.

6 Experimental results with a small prototype

Like [9], we have obtained experimental results for the probability of undetected error for each error-detecting ECSM scheme presented in Section 3. Additionally, experimental results for the probability of giving an incorrect result are

given for each fault-tolerant ECSM scheme discussed in Section 4. These experiments are based on an *exhaustive* generation of faults for a small ECSM module which is modeled using VHDL with a Xilinx Spartan 3 1000 FPGA as the hardware target.

For having an exhaustive fault generation with all the possible ECSM inputs in a reasonable amount of time, we have selected the prototype to be based on the finite field $GF(2^{11})$. For this small prototype, the simplest experiment that involves only the PV module, the number of ECSM operations is approximately 0.5 billion requiring roughly 2.5 hours to complete in the actual hardware. Other experiments involve much more operations, for example in each parallel computing scheme the number of ECSM operations is approximately 35 billions and about 7 days to complete. Although this ECSM prototype is small for a real application, it allows us to perform experiments in an exhaustive way, which in turn permits us to illustrate the error detection coverage for the schemes presented earlier.

6.1 Parameters, fault model, and process

The elliptic curve selected is E: $y^2 + xy = x^3 + x^2 + 1$, over the finite field $GF(2^{11})$ where the trinomial $g^{11} + g^2 + 1$ was used as the reduction polynomial. E has an order of $1982 = 2 \times 991$. The double-and-add algorithm has been used for obtaining the ECSM with the Lopez and Dahab projective coordinates. In our experiments, the finite field operations over $GF(2^{11})$ require 134 gates for their implementation (117, 6 and 11 for finite field multiplication, squaring and addition respectively). The fault model used is a permanent stuck-at-0 or stuck-at-1 fault in these gates, and only one gate can be faulty at a time. Faults have been injected in a similar way as described in [25]. The idea is to add a multiplexor in the output of each gate to be tested, such that we can select if the gate will work normally or with a stuck-at-0 or stuck-at-1 fault at its output.

The elliptic curve utilized has 1980 points on it that are of order either 1982 or 991. In our experiments, these 1980 points[1] are input for each value of scalar k in the range of 2 to 990. For each fixed set of input point and scalar, a fault-free computation is performed to obtain kP. Then a stuck-at-0 or stuck-at-1 fault is injected for each individual gate used for the finite field operations and the ECSM operation is repeated. The result is then compared with the fault-free case to determine if the fault has produced an erroneous result.

For the case of PV, the experiment consists of obtaining, after a single fault (stuck-at-0 or 1), the probability that an erroneous result is on the original elliptic curve E. For the re-computation based schemes, after the injected fault, two scalar multiplications with their respective encoding process are performed (at t_0 and at t_1) and then the results are compared. Additionally, in order to show the importance of the choice of the encoding/decoding processes, to each

[1] The remaining two points include the point at infinite and $((00000000000), (00000000001))$. The latter has the order of two. Both of these points are not considered suitable for cryptographic applications.

re-computation scheme presented in Section 3 (i.e., RC, RC_PV, and RC_partial) we include other types of encoding/decoding processes in the experiment.

Similar to the re-computation case, an experiment is performed for each parallel based schemes presented in Section 3 (i.e., PC and PC_PV). In these schemes there are two ECSM modules. In one set of experiments, we inject faults in only one of the two modules at a time. In another set of experiments we consider the case where each of the two ECSM modules has a single fault at the same time (at the same or different location).

For each fault-tolerant scheme presented in Section 4 (i.e., TMR, DMR, and PRC based schemes) we have two or three ECSM modules. In the experiment, a single fault is injected (at the same or different location) in each ECSM module to detect those cases for which these schemes give an incorrect result.

6.2 Results obtained

6.2.1 Encoding schemes

As we stated in Section 3, the encoding/decoding process plays an important role for error detection capability for schemes shown in Figures 2 and 3. To see their relative effectiveness, we ran experiments for different encoding techniques for ECSM re-computation based schemes. Table 4 shows their probabilities of undetected errors. The results of the table imply that the combination of the encoding schemes (9) and (10) i.e., $(k, P) \mapsto (k'', P')$ yields the least probability of undetected errors.

Encoding	Pr(undetected error)	
	Stuck-at-0 fault	**Stuck-at-1 fault**
$(k, -P)$ [1]	$\frac{1}{117}$	$\frac{1}{12,692}$
$(k', -P)$ [2]	$\frac{1}{128}$	$\frac{1}{1,395,753}$
(k, P') [3]	$\frac{1}{421,190}$	$\frac{1}{2,029,025}$
(k'', P') [3, 4]	$\frac{1}{1,097,914}$	$\frac{1}{7,497,185}$

[1] point negation is needed for decoding [2] $k' = \#E - k$

[3] $P' = (\gamma X, \gamma^2 Y, \gamma Z),\ \gamma \in GF^*(2^m)$ [4] $k'' = k + j(\#E)$

Table 4: Probabilities of undetected errors for re-computation based schemes using different encoding in the experiment over $GF(2^{11})$

6.2.2 Error detection schemes

For each error-detecting scheme presented in Section 3 the probability of undetected error was obtained. Table 5 shows these probabilities for the stuck-at-0 and stuck-at-1 fault models. The PV alone gives the worst error detection coverage for these schemes (i.e., $\frac{1}{217}$ for the stuck-at-0 fault model).

Our experiment for scheme RC shows values that are of the same order of the value from Equation (11) if stuck-at-0 and stuck-at-1 results are combined (i.e., $\frac{1}{1,915,339}$ vs. $\frac{1}{q^2} = \frac{1}{4,194,304}$). On the other hand, scheme RC_PV in our experiment had no undetected errors. This particular value is due to the fact that the number of EC scalar multiplications performed (i.e., $989 \times 134 \times 1980 = 262,401,480$) was less than the value expected from Equation (13) (i.e., $q^3 = 8,589,934,592$).

For both parallel computation based schemes, PC and PC_PV, if the faults are confined in only one of the two ECSM modules then there are no undetected errors as shown in the forth row of Table 5. The results for the case where both ECSM modules had a fault each are shown in the last two rows. It can be noticed that the probability of undetected errors is least for the schemes that combine re-computation or parallel computation along with PV (i.e., schemes RC_PV and PC_PV).

Scheme	Pr(undetected error)		Pr($Q = \mathcal{O}, P$|undetected error)	
	Stuck-at-0 fault	**Stuck-at-1 fault**	**Stuck-at-0 fault**	**Stuck-at-1 fault**
ECSM with PV	$\frac{1}{217}$	$\frac{1}{1,547}$	$\frac{536,935}{603,404}$	$\frac{2,030}{7,711}$
ECSM with PV[1]	$\frac{1}{1,974}$	$\frac{1}{2,100}$	NA	NA
RC	$\frac{1}{1,097,914}$	$\frac{1}{7,497,185}$	$\frac{236}{239}$	$\frac{197}{1,400}$
RC_PV[1], PC[2], PC_PV[2]	0	0	NA	NA
RC_partial[3]	$\frac{1}{19,653}$	$\frac{1}{397,980}$	$\frac{28}{81}$	$\frac{20}{81}$
PC[4]	$\frac{1}{1,678,848}$	$\frac{1}{9,806,261}$	$\frac{20,705}{20,944}$	$\frac{370}{3,447}$
PC_PV[4]	$\frac{1}{1,598,263,560}$	$\frac{1}{17,580,899,160}$	NA	NA

[1] modifying the PV module to exclude $\mathcal{O}$ and P as valid outputs [2] injecting faults only to one ECSM module
[3] $l = 3$ [4] injecting faults to both ECSM modules

Table 5: Probabilities of undetected errors for our experiment over $GF(2^{11})$

6.2.3 Fault-tolerant schemes

For each fault-tolerant scheme presented in Section 4 (i.e., TMR, DMR, and PRC based schemes) the probability of giving an incorrect result was obtained. These probabilities for the stuck-at-0 and stuck-at-1 fault models are given in Table 6. These values are of the same order of the values obtained in Equations (14), (15), and (16). For example, the values obtained for the TMR fault-tolerant ECSM scheme for the two fault models are between the value of Equation (14) (i.e., $\frac{1}{559,616}$ and $\frac{1}{3,060,208}$ vs. $\frac{3}{q^2} = \frac{1}{1,398,101}$). Also, the experiment for the DMR based fault-tolerant ECSM the expected value from Equation (15) is close to the results for each fault model (i.e., $\frac{1}{1,073}$ and $\frac{1}{1,161}$ vs. $\frac{2}{q} = \frac{1}{1,048}$).

Fault-tolerant scheme	Pr(incorrect result)	
	Stuck-at-0 fault	**Stuck-at-1 fault**
TMR	$\frac{1}{559,616}$	$\frac{1}{3,060,209}$
DMR	$\frac{1}{1,073}$	$\frac{1}{1,161}$
PRC	$\frac{1}{385,954}$	$\frac{1}{2,539,627}$

Table 6: Probabilities of an incorrect result for fault-tolerant ECSM schemes in our experiment over $GF(2^{11})$

7 Conclusions

In this report we have presented the concept of encoding using input randomization for the purpose of detecting errors and/or tolerating faults during the ECSM operation. The resulting schemes are based on PV, re-computation, and parallel computation. A combination PV with re-computation or parallel computation permits to have a high probability of error detection for ECSM. Also, this combination provides resistance to attacks where fault-induced operations do not leave the original elliptic curve (e.g., the SCF attack). Additionally, experimental results for a small ECSM prototype have been given. These experiments are based on an exhaustive generation of stuck-at faults for obtaining probabilities of undetected errors for the proposed error-detecting schemes.

Traditionally, the concept of having only two modules working in parallel has been associated with error-detecting systems only. However, for ECSM we have shown that with only two ECSM modules along with either PV (i.e., the proposed DMR based scheme) or re-computation (i.e., PRC based scheme), it is possible not only to detect but also correct errors due to faults. These fault-tolerant schemes are more efficient, i.e., it uses one less ECSM module than the TMR based scheme.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., and Rohatgi, P., "The EM side-channel(s)," Cryptographic Hardware and Embedded Systems (CHES 2002), volume 2523 of LNCS, pp. 29–45, Springer-Verlag, 2002. 1
2. Antipa, A., Brown, D., Menezes, A., Struik, R., and Vanstone, S., "Validation of elliptic curve public keys," Proceedings of PKC 2003, LNCS 2567, pp. 211-223, Springer-Verlag, 2003. 2, 4, 6
3. Biehl, I., Meyer, B., and Müller, V., "Differential Fault Attacks on Elliptic Curve Cryptosystems," Advances in Cryptology, Proc. CRYPTO 2000, LNCS 1880, pp. 131-146, Springer-Verlag, 2000. 2, 4, 6
4. Blömer, J., Otto, M., and Seifert, J.P., "Sign Change Fault Attacks On Elliptic Curve Cryptosystems," Cryptology ePrint Archive, Report 2004/227, 2004. 2, 4

5. Boneh, D., DeMillo, R., and Lipton, R., "On the Importance of Eliminating Errors in Cryptographic Computations," Advances in Cryptology, Proc. EUROCRYPT'97, LNCS 1233, pp. 37-51, Springer-Verlag, 1997. 1
6. Ciet, M., and Joye, M., "Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults," Cryptology ePrint Archive, Report 2003/028, 2003. 2, 4
7. Coron, J., "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems," Cryptographic Hardware and Embedded Systems (CHESS 1999), volume 1717 of LNCS, pp. 292–302, Springer-Verlag, 1999. 7, 8
8. Diffie, W., and Hellman, M. "New Directions in Cryptography," Proceedings of the AFIPS National Computer Conference, 1976.
9. Dominguez-Oviedo, A., and Hasan, M. A., "Error-Detecting and Fault-tolerant Structures for ECC", Technical Report CACR 2005-10, University of Waterloo, 2005. 2, 7, 8, 16, 18
10. Ebeid, N., and Hasan, M. A., "On randomizing private keys to counteract DPA attacks." Selected Areas in Cryptography SAC 03, LNCS 3006, pp. 58-72, Springer-Verlag, 2003. 8
11. FIPS 186-2, Digital Signature Standard. Federal Information Processing Standards Publication 186-2, U.S. Department of Commerce/NIST, 2000. 6, 8
12. Hankerson, D., and Lopez, J., and Menezes, A., "Software implementation of elliptic curve cryptography over binary fields," Cryptographic Hardware and Embedded Systems (CHES 2000), volume 1965 of LNCS, pp. 1–24, Springer-Verlag, 2000. 18
13. Hankerson D., Menezes, A., and Vanstone, S., "Guide to Elliptic Curve Cryptography," Springer, 2003. 6
14. Johnson, B., "Fault-tolerant microprocessor-based systems," IEEE Micro, Vol 4, No 6, pp. 6–21, December 1984. 5
15. Joye, M., and Tymen, C., "Protections against differential analysis for elliptic curve cryptography an algebraic approach," Cryptographic Hardware and Embedded Systems (CHES 2001), volume of 2162 LNCS, pp. 377–390, 2001. 7
16. Kocher, P., Jaffe, J., and Jun, B. "Differential power analysis," Advances in Cryptology, Proc. CRYPTO'99, volume 1666 of LNCS, pp. 388–397, Springer-Verlag, 1999. 1
17. Kocher, P., "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," Advances in Cryptology, Proc. CRYPTO'96, volume 1109 of LNCS, pp. 104–113, Springer-Verlag, 1996. 1
18. Lopez, J., and Dahab, R., "Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$," Selected Areas in Cryptography - SAC '98, Springer-Verlag, LNCS 1556, pp. 201–212, 1999. 3
19. Lutz, J., and Hasan, M. A., "High Performance Elliptic Curve Cryptographic Co-processor," Technical Report, CACR 2004-06, University of Waterloo, 2004. 16
20. Montgomery, P.L., "Speeding the Pollard and elliptic curves methods of factorization," Mathematics of Computation, 48, pp. 243–264, January 1987. 2

21. Patel, J., and Fung, L., "Concurrent error detection in ALUs by recomputing with shifted operands," IEEE Transactions on Computers, Vol. C-31, No. 7, pp. 589–595, July 1982. 5

22. Raynolds, D., and Metze, G., "Fault detection capabilities of alternating logic," IEEE Transactions on Computers, Vol. C-27, No. 12, pp. 1093–1098, December 1978. 5

23. "The Reliability Design Handbook," Rome Air Development Center, Grifiss Air Force Base, N.Y., 1976. 12

24. Standard Specifications for Public Key Cryptography, IEEE, P1363-2000. 3

25. Zarandi, H.R., Miremadi, S.G., and Ejlali, A., "Dependability analysis using a fault injection tool based on synthesizability of HDL models," Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on, Vol., Iss., 3-5, pp. 485–492, Nov. 2003. 19