Error-Detecting and Fault-Tolerant Structures for ECC

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

Abstract

For constrained devices, elliptic curve cryptography (ECC) is an attractive choice because it achieves the same level of security with a much smaller key size in comparison with other schemes such as those that are based on integer factorization or discrete logarithm. For security reasons, especially to provide resistance against fault-based attacks, it is very important to verify the correctness of computations in ECC applications. In this report, fault-tolerant and error-detecting elliptic curve cryptosystems are considered. 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. This will result in more reliable systems where faults may occur due to natural causes. For the purpose of detecting errors due to faults, a number of schemes based on the point-on-the-curve checking, time redundancy, and hardware redundancy are presented. A combination of the point-on-the-curve checking and time or hardware redundancy can be used for detecting errors with a very high probability during the computation of the elliptic curve scalar multiplication (ECSM). Additionally, we show that using dual modular redundancy (DMR) and the point-on-the-curve checking, it is possible to have a fault-tolerant structure for the ECSM. If certain conditions are met, this scheme is more efficient than others such as the well-known triple modular redundancy.

1 Introduction

Cryptography refers to design principles, means and methods for rendering plain information unintelligible to unauthorized parties. In the past, cryptography was used only for secret communications between powerful security entities such as military and intelligence agencies. Today, with the widespread use of computers and the Internet, secure communications are more than a privilege; they are a priority requirement even for the general public. In 1976, with the introduction of public key cryptography by Diffie and Hellman [5], secure communications were made practical. E-commerce and smart cards are examples of how cryptographic applications have become a part of everyday life.

Unfortunately, cryptosystems have been continuously subject to attacks. Even when attacks are mounted for research purposes, they constitute a risk for protocols and systems in terms of security. Cryptoanalytic attacks may reveal system's vulnerabilities which then need to be fixed with countermeasures. In [12], Kocker, Jaffe and Jun introduced the significance of cryptographic applications being resistant to side-channel analysis (e.g., reconstruction of a secret key from analysis of the power consumed during cryptographic operations).

Another type of attacks that should be taken into account is the fault analysis attack. Introduced by Boneh, DeMillo, and Lipton [3], this attack is based on producing malfunctions in cryptosystems to leak sensitive information (i.e., secret keys). They have showed how some cryptographic schemes such as the RSA and Rabin digital signatures are vulnerable to induced computational errors. Particularly, for an implementation of RSA based on the Chinese Remainder Theorem, they have demonstrated that with only two signatures of the same message (one computed correctly and one produced after some fault) it is possible to efficiently factor the modulus used. In order to avoid such attacks, they suggest verifying the correctness of computations in cryptographic applications.

Biehl, Meyer, and Muller extended fault-based attacks to cryptosystems using elliptic curves [2]. Their basic idea is to enforce, by a fault, a computation in a weaker group where solving the elliptic curve discrete logarithm problem (ECDLP) is feasible. Using this principle, they show that it is possible to derive secret information (e.g., a secret key) from a tamper-resistant device computing the elliptic curve (EC) scalar multiplication. They assume that it is possible for an attacker to select the input point P or to induce an error in that point. Specifically, they have based their work on the EC ElGamal cryptosystem where it is necessary to provide the scalar multiplication oracle to an external entity (e.g., smart card terminal reader) for encryption. In this way, an adversary can select an input point and mount this attack.

As discussed above, the fault-based attacks against cryptosystems are a real threat and should be taken into account. Accordingly, the design of cryptosystems should include some countermeasures against fault-based attacks. Elliptic curve cryptosystems are not the exception as a leading cryptographic scheme. In this report, we present our work on error-detecting and fault-tolerant elliptic curve cryptosystems. Error detection may be a sufficient countermeasure for many security applications. However, fault-tolerant characteristic enables a sys-

tem to perform its normal operation in spite of faults. This will result in more reliable systems where faults may occur due to natural causes such as, abnormal temperature, electromagnetic interference (EMI), or power supply changes.

The organization of this report is as follows. In Section 2, we give a brief overview of elliptic curve cryptography (ECC) and existing fault-based attacks on ECC. Section 3 presents error-detecting structures for ECC. Experimental results for the probability of undetected errors are presented in Section 4. In Section 5, fault-tolerant ECC structures are given. Finally, overhead costs based on implementation results are given in Section 6 while some concluding remarks are made in Section 7.

2 Background

2.1 Elliptic curve cryptography (ECC) overview

Let F be a finite field. An *elliptic curve* E over F is formed by the points (X, Y, Z) that satisfy the following equation:

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, \quad a_i \in F. \quad (1)$$

Equation (1) is referred to as the *projective* form of the Weisterass equation. It is very common to use an alternative representation called the *affine* form of the Weisterass equation, which can be obtained from Equation (1) by performing a change of variables $y = (Y/Z)$ and $x = (X/Z)$ as given below

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in F. \quad (2)$$

For binary finite fields, from Equation (2), one can perform an admissible change of variables [9] that transforms E to the curve:

$$y^2 + xy = x^3 + ax^2 + b. \quad (3)$$

The points of an elliptic curve with the point at infinity ($\mathcal{O}$) form an abelian group under a particular operation, denoted as point addition ($\uplus$). Let E be a curve over the binary field $GF(2^m)$ which satisfies Equation (3). Let P and Q be any two distinct points on E. The rules for point addition are given as follows:

1. The point $\mathcal{O}$ is used as the *identity element*. For any point P, $P \uplus \mathcal{O} = P$ and $\mathcal{O} \uplus P = P$.

2. The *negative* of the point $P = (x, y)$, denoted as $-P$, such that $P \uplus (-P) = \mathcal{O}$ is $-P = (x, x + y)$.

3. Let $P = (x_1, y_1) \neq \mathcal{O}$ and $Q = (x_2, y_2) \neq \mathcal{O}$, where $P \neq \pm Q$. The *point addition*, $R = P \uplus Q = (x_3, y_3)$, is defined as follows,
   $x_3 = \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a$, and $y_3 = \left(\frac{y_1+y_2}{x_1+x_2}\right)(x_1 + x_3) + x_3 + y_1$.

4. Let $P = (x_1, y_1) \neq \mathcal{O}$ and $P \neq -P$. The *point doubling*, $R = P \uplus P = 2P = (x_3, y_3)$, is defined as follows,

$$x_3 = \left(x_1 + \frac{y_1}{x_1}\right)^2 + \left(x_1 + \frac{y_1}{x_1}\right) + a, \text{ and } y_3 = (x_1 + x_3)\left(x_1 + \frac{y_1}{x_1}\right) + x_3 + y_1.$$

The *order of an elliptic curve* E, denoted as $\#E$, is defined as the number of points on the curve. Additionally, the *order of a point* P, denoted as $ord(P)$, on an elliptic curve E is the smallest positive integer e such that by adding this point e times the infinite point $\mathcal{O}$ is obtained.

Using a result of Lagrange's theorem from group theory [20], we can affirm that the order of any point always divides the order of the group $\#E$. As a result, if $\#E$ is prime, then the order of any point is $\#E$ and the point can be used as a generator. For cryptographic applications, the order of a selected point P should be a factor of a sufficiently large prime [18].

Theorem 1 *(*Hasse's Theorem*). Let E be an elliptic curve over $GF(q)$. Then*

$$\#E(GF(q)) = q + 1 - t, \quad \textit{where } |t| \leq 2\sqrt{q}.$$

The above addition rules are for affine coordinates (i.e., (x, y)). In order to reduce computational cost, a number of projective coordinate systems (i.e., (X, Y, Z)) have been suggested in the literature. Among these projective systems, for the binary field $GF(2^m)$, the one by Lopez and Dahab [13] appears to be the most efficient, and is related to the affine system as follows:

$$x = \frac{X}{Z} \text{ and } y = \frac{Y}{Z^2}.$$

Let k be a positive integer and P be a point on E, then the *elliptic curve scalar multiplication (ECSM)* is given as follows

$$kP = \underbrace{P \uplus P \uplus \cdots \uplus P}_{k \text{ times}}.$$

To carry out ECSM, it is necessary to perform point doublings and additions. As described in the EC group law above, such operations require a number of finite field arithmetic operations (i.e., multiplication, addition, squaring, and multiplicative inverse).

Let $(k_{t-1}\ k_{t-2} \cdots k_1\ k_0)_2$ be the t-bit binary representation of k. The scalar multiplication kP can be computed as follows,

$$Q = kP = \left(\sum_{i=0}^{t-1} k_i 2^i\right) P,$$

which can be written as

$$Q = 2(2(\cdots 2\,(2\,(k_{t-1}P) \uplus k_{t-2}P) \uplus \cdots) \uplus k_1 P) \uplus k_0 P.$$

This operation can be computed using the following well-known double-and-add algorithm.

Algorithm 1 *ECSM by double-and-add*
Input: P *and* $k = (k_{t-1}\ k_{t-2} \cdots k_1\ k_0)_2$
Output: $Q = kP$
$Q \leftarrow \mathcal{O}$
for $i = t-1$ *to 0 by -1 do*
$Q \leftarrow 2Q$
if $k_i = 1$ *then*
$Q \leftarrow Q \uplus P$
Return Q.

Assuming that k has approximately the same size as the dimension of the underlying finite field $GF(2^m)$ (i.e., $t \approx m$), this algorithm would require approximately m point doublings and $m/2$ point additions. In Table 1, the number of finite field operations over $GF(2^m)$ required for ECSM are provided for affine and the Lopez and Dahab projective coordinates. The symbols M, S, A, and I denote, the cost of finite field multiplication, squaring, addition and inversion respectively.

Coordinate system	**No. of finite field operations required**		
	Point addition	**Point doubling**	**ECSM**
Affine (x, y)	$2M + 1S + 8A + 1I$	$2M + 1S + 6A + 1I$	$(3M + \frac{3}{2}S + 10A + \frac{3}{2}I)m$
Lopez and Dahab[1] (X, Y, Z)	$10M + 4S + 8A$	$5M + 5S + 4A$	$(10M+7S+8A)m+1I$

[1] using mixed coordinates for point addition

Table 1: Cost of point addition, point doubling, and ECSM using double-and-add algorithm

2.2 Recent ECC fault attacks and countermeasures

In 2000, Biehl, Meyer, and Muller proposed the first fault-based attack on ECC [2]. Their basic idea is to force, through a fault, a computation in a weaker group where solving the ECDLP is feasible. They assume that the attacker can select the input P of the ECSM (e.g., in an EC ElGamal cryptosystem). From Equation (2) they observe that the elliptic curve parameter a_6 is not used for the point addition using affine coordinates. Then, an appropriate finite field pair element $\widetilde{P} = (x, y)$ is selected that is not over the original elliptic curve $E(a_1, a_2, a_3, a_4, a_6)$ but is a point of a cryptographically less secure elliptic curve $\widetilde{E}(a_1, a_2, a_3, a_4, \widetilde{a}_6)$. The relation between $\widetilde{a}_6$ and $\widetilde{P}$ can be obtained as follows,

$$\widetilde{a}_6 = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x.$$

In this way the order of $\widetilde{E}$, $\#\widetilde{E}$, could have a small divisor r such that $ord(\widetilde{P}) = r$. Repeating this procedure with sufficiently different chosen points $\widetilde{P}_i$ supplies the values of $k \bmod r_i$ from $k\widetilde{P}_i$, where $ord_{\widetilde{E}_i}(\widetilde{P}_i) = r_i$. Finally, the

value of k can be obtained using the Chinese remainder theorem (CRT). They conclude that it is important to check whether the input point is on the curve or not. Also, if it is possible to produce an error in P just after the point on the curve-checking process, then a point-on-the-curve checker at the end of the computation will help to avoid this attack.

The same idea can be applied to the reduced elliptic curve Equation (3). For point addition, parameter b is not used. Then this attack can be mounted selecting a point $\widetilde{P} \in \widetilde{E}(a, \widetilde{b})$, where $\widetilde{E}$ is a less secure elliptic curve than E with different value of b.

Recently in [4], Ciet and Joye have showed how to recover the secret key k by having an unknown but fixed faulty input point $\widetilde{P}$. This is a more realistic scheme because, excluding EC ElGamal encryption, in most EC cryptosystems the point P is predetermined and might be fixed. The attacker cannot select this point which is usually stored in the memory of a tamper-proof device. Their fault model considers that only one coordinate x or y is faulty. If it is assumed that the x-coordinate of a point $P = (x, y)$ is altered, the resultant point will be $\widetilde{P} = (\widetilde{x}, y)$. The scalar multiplication algorithm computes $\widetilde{Q} = k\widetilde{P}$. The faulty result $\widetilde{Q} = k(\widetilde{x}, y) = (\widetilde{x}_k, \widetilde{y}_k)$, is over a curve $\widetilde{E}$ on which the operation was performed, defined as $\widetilde{E}(a_1, a_2, a_3, a_4, \widetilde{a}_6)$, where $\widetilde{a}_6$ is obtained as follows

$$\widetilde{a}_6 = \widetilde{y}_k^2 + a_1\widetilde{x}_k\widetilde{y}_k + a_3\widetilde{y}_k - \widetilde{x}_k^3 - a_2\widetilde{x}_k^2 - a_4\widetilde{x}_k.$$

Using Equation (2) for $\widetilde{E}$ we obtain

$$\widetilde{E} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + \widetilde{a}_6. \tag{4}$$

Since $P \in \widetilde{E}$, it needs to follow Equation (4). Substituting its coordinates $(\widetilde{x}, y)$ and grouping the terms with $\widetilde{x}$ we have the following expression

$$\widetilde{x}^3 + a_2\widetilde{x}^2 + (a_4 - a_1y)\widetilde{x} + (\widetilde{a}_6 - y^2 - a_3y) = 0$$

By solving the above equation for $\widetilde{x}$, we can have up to three different roots, where one is the fault generated x-coordinate. If the point P is known (usually this parameter is public), $\widetilde{x}$ can be distinguished easily since it will differ from x just for the faulty bits. Having $\widetilde{P}$ and $\widetilde{Q}$, the procedure described in [2] for obtaining k can be followed. Additionally, the authors consider other attacks where the underlying finite field or the elliptic curve parameters are disturbed by a fault. They assume that it is possible to inject a transient fault into these parameters. For example, in a smart card system parameters are usually stored in non-volatile memory (e.g., EEPROM) and they are transferred into RAM for executing a specific algorithm. It is possible to induce an error in such data before or during this transfer.

Antipa, Brown, Menezes, Struik and Vanstone state the importance of checking whether the received point for any EC key agreement and public-key encryption protocols are on the proper elliptic curve [1]. Specifically, they show the vulnerability under this scenario of the one-pass EC Diffie-Hellman protocol,

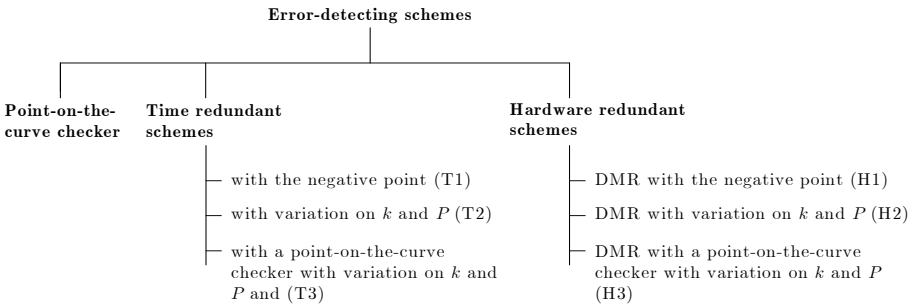

Figure 1: Classification of proposed error-detecting schemes

the EC integrated encryption scheme (ECIES), the one-pass ECMQV protocol and the EC digital signature algorithm. Some of these protocols or algorithms are defined in standards such as ANSI X9.62, ANSI X9.63, IEEE 1363-2000, ISO/IEC 15946-3, FIPS 186-2, and ISO 15946-2. They mention the significance for implementers of taking into account point verification, even when some standards do not mandate that. A possible countermeasure considered in [1] is to use other formulas for the addition law that use both elliptic curve parameters, a and b. Such formulas usually require more computations and hence cause a degradation in performance.

3 Error-detecting structures for ECC

For having error-detecting and/or fault-tolerant capabilities, it might be necessary to add some kind of "redundancy" in the design. The redundancy added to a system could be extra hardware, information, time or any combination of these. Hardware redundancy uses some extra components (e.g., modules, gates, memory registers, and buses). Information redundancy can be carried out by utilizing some error detection and/or correction techniques based on coding theory. Time redundancy involves extra computation of the same operation, possibly implemented by different methods and comparing the results for detecting an error. In this work, we mainly use the concepts of hardware redundancy and time redundancy.

In this section, error-detecting schemes and structures are proposed. Here, we focus on a high-level design, where the ECSM module is the main block. Schemes presented in this section rely on the point-on-the-curve checker, time redundancy, and hardware redundancy. They are summarized in Figure 1. As shown in this figure and also in related mathematical expressions that are to be presented later in this section, the three time redundant schemes are labelled as T1, T2, and T3, for compactness. Similarly, the hardware redundant schemes are labelled as H1, H2, and H3.

For each of the schemes presented here, we give its reliability, which is defined as its characteristic expressed by the probability that it will perform its function [17]. Throughout the rest of the document, we assume that 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 [2] and [1]. Additionally, we assume that an appropriate elliptic curve has been selected (e.g., using the guidelines of a recognized standard such as FIPS 186-2 [7]). For our analysis we will deal with the binary finite field $GF(2^m)$; however, these concepts can in principle be extended to the prime field $GF(p)$.

3.1 Error detection using a point-on-the-curve checker

A point-on-the-curve checker is a computational module that takes a point $P = (x, y)$ as its input and checks whether the point is on the elliptic curve E. This is done simply by verifying whether the coordinates of P satisfy Equation (3). If they do, only then an affirmative signal (say 'ok') is generated as an output. A point-on-the-curve checker can be implemented either in hardware or in software. The work presented here assumes a hardware implementation.

A point-on-the-curve checker can be used for error-detection proposes as shown in Figure 2. In ECC, each point can be represented as a pair of finite field elements. For example in affine coordinates, a point on an elliptic curve defined over the binary field of dimension m has the form (x, y) where x, $y \in GF(2^m)$. For this field we can have 2^{2m} different finite field pairs in total. However, based on Hasse's theorem pertaining to the order of the elliptic curve, only about 2^m of these pairs are points on the curve.

An error produced by a fault in any ECSM operation may result in a finite field pair that does not correspond to a valid point of the curve. Simply stated, assume this error produces a random pair of finite field elements $(\widetilde{x}, \widetilde{y}) = \widetilde{Q}$. In such a case, the probability that this finite field pair satisfies the elliptic curve equation can be obtained as follows:

$$\Pr(\widetilde{Q} \in E) = \frac{\#E}{\text{Number of finite field pairs}}, \tag{5}$$

$$\Pr(\widetilde{Q} \in E) \approx \frac{2^m}{2^{2m}} = \frac{1}{2^m}. \tag{6}$$

Based on the NIST recommendations for ECC applications, m should be at least 163 [6]. In such a case, the value obtained in (6), which represents the probability of having an incorrect result point as part of the original elliptic curve, is quite small. Additionally, the incorrectly computed point $\widetilde{Q} \in E$ is not a great risk under the EC fault-based attacks described in Section 2. The principal reason is that those attacks use input points over a less secure elliptic curve $\widetilde{E}$, where the ECDLP is easier.

Under this fault model (i.e., a fault produces a random finite field pair) we can state that the ECSM is a nearly self-checking operation using the point-on-the-curve checking process for every result. This scheme can be implemented

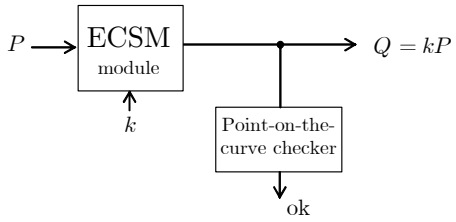

Figure 2: Point-on-the-curve checker after the ECSM module

easily since the point-on-the-curve checker requires only three finite field multiplications, two finite field squarings, and three finite field additions (see Equation (3)). The finite field multiplier in this module could be designed to run at a much slower speed since this checking is done just once for each scalar multiplication. On the contrary, the multiplier for the ECSM is often optimized for speed since its operation is repeated many times. For instance, the ECSM using Algorithm 1 with affine coordinates over $GF(2^{163})$ will need approximately 489 finite field multiplications, 244 squarings, 1630 additions and 244 inverses.

Let r_{ECSM} and r_{PC} be the reliabilities of the ECSM module and the point-on-the-curve checker respectively. Then the reliability of the point-on-the-curve checker based system shown in Figure 2 is

$$R_{PC} = r_{ECSM}\, r_{PC}.$$

Although, the system reliability is reduced in comparison with a stand-alone ECSM module, the system is able to detect most of the errors if the point-on-the-curve checker, which can be significantly smaller than the ECSM module, is not in error. In fact, the point-on-the-curve checker can be designed such that $r_{PC} \approx 1$. In such a case, $R_{PC} \approx r_{ECSM}$.

3.2 Error detection using time redundancy

For applications where the time required for error detection is not critical, it is possible to recompute the result with the same or different methods and check the results to detect a possible error. A general time redundant scheme is shown in Figure 3. In the bottom data path there are two extra processes: encoding and decoding. This is necessary because if the computation module is permanently in a faulty state, a repetition of the process might reiterate the errors.

Encoding and decoding could be carried out through a number of approaches. For conventional number systems the most popular ones include: alternating logic [16], recompute with shifted operands (RESO) [15] and recompute with swapped operands (RESWO) [11]. Below, we provide other types of encoding and decoding processes that are suitable for ECC.

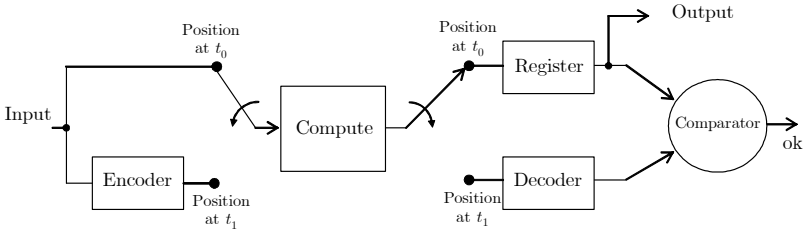

Figure 3: General time redundant scheme

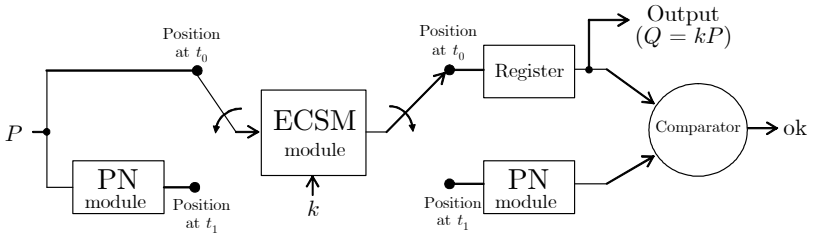

Figure 4: ECSM using time redundancy with the negative point (scheme T1)

3.2.1 Time redundancy with the negative point

Taking advantage of the simplicity of negating an EC point (i.e., $-(x, y) = (x, x + y)$), it is possible to easily perform ECSM twice– first normally at time t_0 with point P, and then at time t_1 using $-P$ as shown in Figure 4. The result of the second ECSM can be inverted to obtain the original result point Q. Then, it is possible to add a comparator to check that both results are equal and, if they are, it is likely that the result is correct. Figure 4 illustrates this scheme for a binary finite field $GF(2^m)$. Note that for obtaining the negative point we only need to perform a simple finite field addition.

In order to obtain the reliability of this system, let us assume that r_{ECSM}, r_{PN}, and r_{Reg} are the reliabilities of the ECSM module, the point negation (PN) module, and the register respectively. Let r_{Comp} be the reliability of the comparator. The system reliability is associated with the probability that all the elements work without error, and it can be obtained as follows:

$$R_{T1} = r_{PN}^2 \, r_{ECSM} \, r_{Reg} r_{Comp}.$$

The point negation (PN) modules in Figure 4 essentially play the roles of the encoder and decoder modules shown in Figure 3. However, these PN modules are not able to detect single faults always. This is discussed in Appendix A with an example.

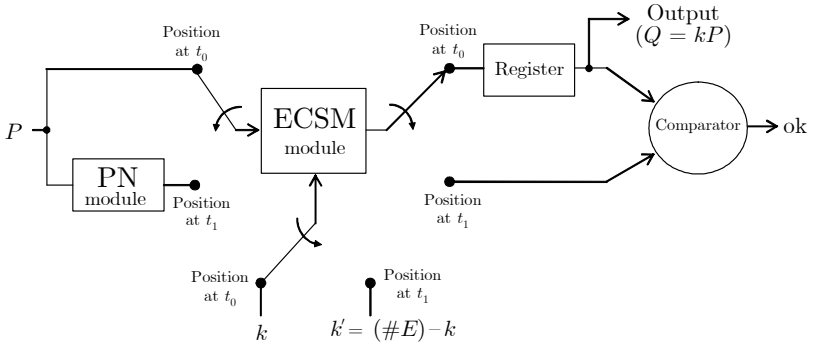

Figure 5: ECSM using time redundancy with variation on k and P (scheme T2)

3.2.2 Time redundancy with variation on k and P

As discussed before, the order of any point always divides the order of the group $\#E$. This property of elliptic curve E can be used to detect errors in ECSM. Towards this, first we compute the ECSM normally with k and P; and then we compute another ECSM with the negative point and a modified scalar, $k' = (\#E) - k$. Note that if there is no error, then $k'P$ is equal to $-Q$, where $Q = kP$. The PN module can be placed before or after the ECSM at t_1. The advantage of having it before the ECSM module is that we have scalars and input points that are different at t_0 and t_1 as illustrated in Figure 5. This scheme appears to be more attractive than the previous time-redundancy structure if the parameters k and k' are predetermined. However, for arbitrary k and k', the operation $k' = (\#E) - k$ may have non-negligible cost. For a binary finite field $GF(2^m)$, it involves an m-bit subtraction. For constrained devices (e.g., 8-bit processor smart card), the cost may not be ignored. An alternative solution for varying the parameter k without this subtraction is to use a different representation of k. It can be implemented using the binary signed digit representation (BSD).

Similar to the previous scheme the system reliability can be obtained as follows

$$R_{T2} = r_{PN}\, r_{ECSM}\, r_{Reg} r_{Comp}.$$

3.2.3 Time redundancy with a point-on-the-curve checker with variation on k and P

If the previous scheme of varying k and P is used along with a point-on-the-curve checker, it is possible to take advantage of both schemes. The resultant structure is shown in Figure 6. In this case, an output Q is considered to be valid if both the point-on-the-curve checker and the comparator detect no error. Note that the point-on-the-curve checker could be placed on either of the two

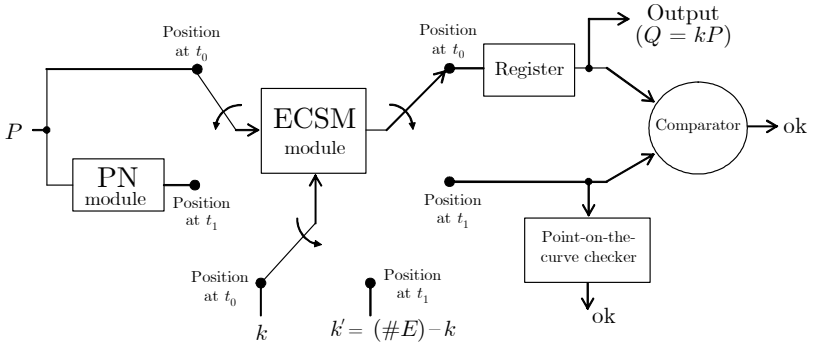

Figure 6: Time redundancy with a point-on-the-curve checker with variation on k and P (scheme T3)

paths connected to the comparator inputs. This structure can considerably reduce the probability of error as we will see in Section 4. The reliability of this time redundancy based scheme can be expressed as follows

$$R_{T3} = r_{PN}\, r_{ECSM}\, r_{Reg} r_{Comp} r_{PC}.$$

3.3 Error detection using hardware redundancy

The schemes discussed in the previous subsection rely on time redundancy. In this subsection, their counterparts that use hardware redundancy are presented. In particular, dual modular redundancy (DMR) is used where we have two independent ECSM modules working in parallel. A DMR based scheme can detect any error in one module. Since DMR has two distinct ECSM modules, the likelihood of having the same permanent or transient fault is low.

3.3.1 DMR with the negative point

Figure 7 shows the scheme for computing the ECSM using two independent modules, one normally with point P and the other with the negative of this point. The reliability of this DMR based scheme can be obtained as follows

$$R_{H1} = r_{PN}^2\, r_{ECSM}^2\, r_{Comp}.$$

It is clear that $R_{H1} < r_{ECSM}$. In other words, the reliability of this system is not better than the reliability of a stand-alone ECSM module. The advantage of this system is that any error associated with a permanent or transient fault in only one module will be detected.

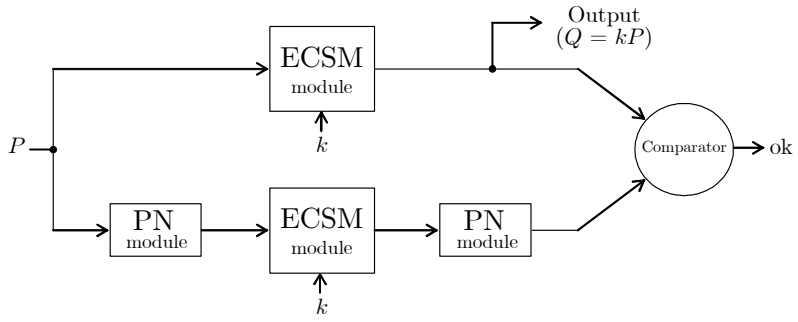

Figure 7: DMR based ECSM with variation on P (scheme H1)

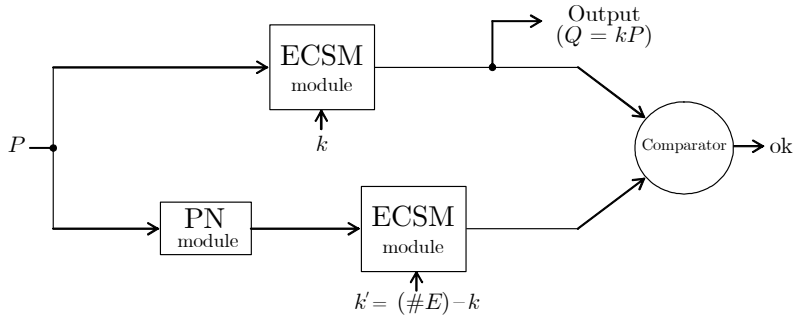

Figure 8: DMR based ECSM with variation on k and P (scheme H2)

3.3.2 DMR with variation on k and P

As mentioned above, if the parameter k is predetermined, an attractive solution is to compute one module with a variation on k and P as illustrated in Figure 8. Similar to the previous scheme the system reliability can be obtained as follows

$$R_{H2} = r_{PN}\, r_{ECSM}^2\, r_{Comp}.$$

3.3.3 DMR with a point-on-the-curve checker with variation on k and P

Similar to the time redundancy case, in DMR we can add a the point-on-the-curve checker to reduce the probability of undetected errors. This is shown in Figure 9 and its reliability can be expressed as follows

$$R_{H3} = r_{PN}\, r_{ECSM}^2\, r_{Comp} r_{PC}.$$

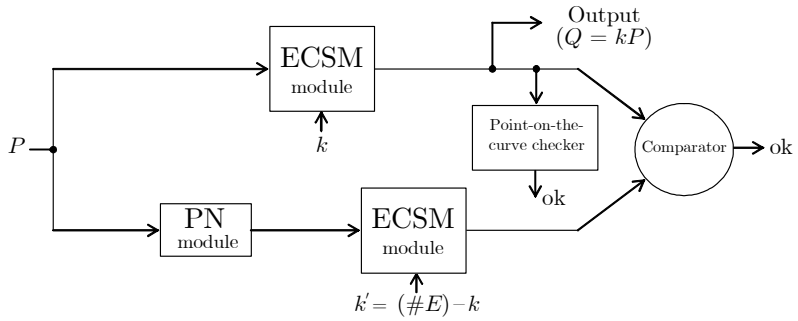

Figure 9: DMR with a point-on-the-curve checker with variation on k and P (scheme H3)

3.4 Remarks for error-detecting structures

For error detection the ECSM could be considered as a nearly self checking operation using the point-on-the-curve checking process. However, if this concept is used in conjunction with some kind of redundancy, hardware or time, it is possible to have even a better error checking capability. The penalty for adding the redundancy is a lower system reliability in comparison with the stand-alone ECSM module. A way to increase the system reliability is to use some fault tolerant techniques to be discussed in Section 5.

It should be noted that compared to the ECSM module, each of the point-on-the-curve checker, the PN module, the register, and the comparator 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 the reliabilities of the proposed time and hardware redundant schemes are given as follows:

$$R_{Tn} \approx r_{ECSM} \qquad n = 1, 2, 3.$$

$$R_{Hn} \approx r_{ECSM}^2 \qquad n = 1, 2, 3.$$

4 Experimental results for undetected errors

As discussed earlier, even though an error may occur due to a fault in the ECSM module, it is possible that the error remains undetected by the schemes presented in the previous section. Whether a fault causes an error or not depends on a number of factors including the input pair k and P, and the fault itself (i.e., its type, location, etc.). The latter, in turn, depends on the implementation of the ECSM module. In this section, we present experimental results of undetected error probabilities based on exhaustive generation of faults for a small prototype ECSM module. The prototype is modelled using VHDL with a Xilinx Spartan 3 1000 FPGA as the hardware target.

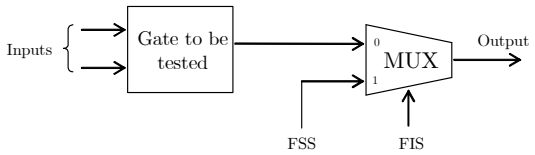

Figure 10: Stuck-at-0 or 1 fault injection method

4.1 Parameters and fault model

For having an exhaustive fault generation with all the possible ECSM inputs in a reasonable amount of time, we have selected the finite field $GF(2^{11})$. For efficient implementation the irreducible binary trinomial $g^{11} + g + 1$ of degree 11 has been selected to construct the finite field. The elliptic curve used is E: $y^2 + xy = x^3 + x^2 + 1$, which has an order of $1982 = 2 \times 991$, where 991 is a prime number. Algorithm 1 has been used for obtaining the ECSM with the Lopez and Dahab projective coordinates. A final point conversion to affine coordinates which includes a multiplicative inverse operation is needed at the end of the ECSM computation. For finite field operations, the methods illustrated in Table 2 have been utilized. In total 134 gates have been used for implementing the finite field operations using these particular methods over $GF(2^{11})$. 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 [21]. The idea is to add a multiplexor to 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 (see Figure 10). Presence or absence of a fault is controlled by the *fault injection signal* (FIS). The *fault selector signal* (FSS) chooses a stuck-at-0 or stuck-at-1 fault for the experiment.

Finite field operation	**Method**	**Number of gates**
Multiplication	Look-up table-based group level Multiplication [8] [1]	117
Squaring	Modified look-up table-based group level Multiplication [14] [1]	6
Inversion	Itoh & Tsujii [10]. With 4 multiplications and 10 squarings.	0
Addition	Bit-wise XOR	11

[1] group size = 6

Table 2: Methods utilized for finite field operations over $GF(2^{11})$

As stated earlier, the order of the curve E is $1982 = 2 \times 991$ where 991 is

prime. This curve has 1980 points on it that are of order either 1982 or 991. In our experiments, these 1980 points 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 Q. Then a stuck-at-0 or stuck-at-1 fault is injected for each individual gate used for the finite field operations. This result is then compared with the fault-free case to determine if the fault has produced an erroneous result $\widetilde{Q}$.

4.2 Results for the point-on-the-curve checker

As we have discussed, for the purpose of error detection one can use a point-on-the-curve checker placed after ECSM module. For this scheme, the probability of undetected errors can be expressed as the probability of an erroneous result $\widetilde{Q}$ that is on the original elliptic curve E. Table 3 shows these probabilities for the stuck-at-0 and stuck-at-1 fault models for our experiment where the underlying field is $GF(2^{11})$. We have noticed that for both types of faults and especially for the stuck-at-0 fault model, the probability of obtaining two special points, namely $\mathcal{O}$ and P is relatively high. The following two scenarios contribute to these higher probabilities. Firstly, if the projective Z-coordinate of the final result ($\widetilde{Q}_Z$) before the conversion to affine coordinates is zero then, independent of the values of X and Y coordinates the result will be the $\mathcal{O}$ point. Secondly, consider the case where the Z-coordinate of $\widetilde{Q}$ in the last iteration of the loop in Algorithm 1 is zero. In such a case, the final result will be $\mathcal{O}$ or P depending of the value of the least significant bit of k.

Fault	$\mathbf{Pr}(\widetilde{Q} \in E)$	$\mathbf{Pr}(\widetilde{Q} = \mathcal{O})$	$\mathbf{Pr}(\widetilde{Q} = P)$	$\mathbf{Pr}(\widetilde{Q} \in E, \widetilde{Q} \neq \mathcal{O}, P)$
Stuck-at-0	$\frac{1}{79.8}$	$\frac{1}{90.0}$	$\frac{1}{1{,}106.0}$	$\frac{1}{1{,}950.4}$
Stuck-at-1	$\frac{1}{1{,}540.6}$	$\frac{1}{6{,}906.4}$	$\frac{1}{38{,}374.0}$	$\frac{1}{2{,}091.0}$

Table 3: Probabilities of erroneous results that are part of E in the experiment over $GF(2^{11})$

With the above observations, it is useful that the point-on-the-curve checker does not consider $\mathcal{O}$ or P as a valid output of the ECSM. In fact, from the cryptographic point of view, if P and k are appropriately selected (i.e., $2 \leq k \leq ord(P) - 1$), a valid result cannot be either $\mathcal{O}$ or P. If we assume that the point on the curve checker is modified such that $\mathcal{O}$ and P are not considered as a valid ECSM output, the resulting probabilities are very close to the value obtained using Equation (5) i.e., $\frac{1982}{2^{11}2^{11}} = \frac{1}{2116.2}$ as shown in the right-most column of Table 3.

Another interesting observation of this experiment is that it is more likely to obtain a result equal to $\mathcal{O}$ with faults that are stuck-at-0 than those with stuck-at-1. For our experiments, stuck-at-0 faults tend to reduce the Hamming weight of the Z-coordinate of the output before it is being converted to affine coordinates as shown in Table 4. In contrast, on average stuck-at-1 faults pro-

duce results with more binary 1s. For this reason, the stuck-at-0 faults have more cases with $\widetilde{Q}_Z = 0$, and consequently the resulting point on the curve after the affine conversion corresponds to $\widetilde{Q} = \mathcal{O}$.

Case	**Average**$(H(\widetilde{Q}_Z))$
Fault free	5.5
Stuck-at-0	5.159
Stuck-at-1	5.534

Table 4: Average Hamming weight the Z-coordinate of the result in the projective system for $GF(2^{11})$

4.3 Results for time redundant schemes

In our experiments with each of the time redundant schemes, after injecting a single fault (stuck-at-0 or 1), two scalar multiplications are performed (at t_0 and at t_1) and then the results are compared. This process is repeated for all single faults and input pair (k, P) combinations. Note that our ECSM implementation has 134 gates that are used for finite field operations and hence there are $134 \times 2 = 268$ cases of single stuck-at faults. The probability of undetected error for each time redundant scheme is shown Table 5. It can be noticed from the table that the probability of undetected error is least for the scheme shown in Figure 6 which uses the point-on-the-curve checker and two different combinations of k and P. Additionally, it can be noticed that for a given redundant scheme the probability of undetected error with a stuck-at-0 fault is higher than that with stuck-at-1 faults. This is because most of the undetected errors in the stuck-at-0 fault case are equal to $\mathcal{O}$.

Time redundant scheme	Pr(undetected error)		Pr($\widetilde{Q} = \mathcal{O}$|undetected error)	
	Stuck-at-0 fault	**Stuck-at-1 fault**	**Stuck-at-0 fault**	**Stuck-at-1 fault**
Negative point (Fig.4)	$\frac{1}{117.3}$	$\frac{1}{12,692.3}$	0.936262	0.138918
Variation on k and P (Fig.5)	$\frac{1}{128.0}$	$\frac{1}{1,395,752.6}$	0.997544	0.180851
With a point-on-the-curve checker[1] with variation on k and P (Fig.6)	$\frac{1}{52,480,296}$	$\frac{1}{131,200,740}$	0	0

[1] modified to exclude $\mathcal{O}$ and P as valid outputs

Table 5: Probabilities of undetected errors for time redundant schemes in the experiment over $GF(2^{11})$

4.4 Results for hardware redundant schemes

For each of the hardware redundant schemes presented in Section 3, there are two ECSM modules. Each of these modules can have one single fault. Elliptic curve scalar multiplications are performed by both ECSM modules and their results are compared. Table 6 shows probabilities of undetected errors for our three hardware redundant schemes. Again, the third scheme (DMR with a point-on-the-curve checker with variation on k and P) has the least probability of undetected errors.

Dual modular redundancy scheme	Pr(undetected error)		Pr($\widetilde{Q} = \mathcal{O}$|undetected error)	
	Stuck-at-0 fault	**Stuck-at-1 fault**	**Stuck-at-0 fault**	**Stuck-at-1 fault**
Negative point (Fig.4)	$\frac{1}{7{,}744.8}$	$\frac{1}{270{,}932.3}$	0.996626	0.482521
Variation on k and P (Fig.5)	$\frac{1}{7{,}959.3}$	$\frac{1}{314{,}839.7}$	0.996863	0.536693
With a point-on-the-curve checker[1] with variation on k and P (Fig.6)	$\frac{1}{416{,}527{,}241.4}$	$\frac{1}{417{,}278{,}538.9}$	0	0

[1] modified to exclude $\mathcal{O}$ and P as valid outputs

Table 6: Probabilities of undetected errors for hardware redundant schemes in the experiment over $GF(2^{11})$

5 Fault-tolerant structures for ECC

From the cryptographic point of view, error detection may be a sufficient countermeasure for avoiding fault-based attacks. However, fault-tolerant characteristic enables a system to perform its normal operation in spite of faults. This will result in more reliable systems where faults may occur due to natural causes such as, abnormal temperature, electromagnetic interference (EMI), or power supply changes.

In this section, methods for fault tolerance for ECSM are presented. First, a classic example of hardware redundancy, triple modular redundancy (TMR), is considered for ECSM. Then, we show that using dual modular redundancy (DMR) and the point-on-the-curve checker, it is possible to have fault-tolerant structure for the ECSM. Finally, we provide a reliability comparison between these two schemes.

5.1 TMR based fault-tolerant ECSM

TMR was proposed by Von Neumann [19] and it utilizes three elements performing the same operation. In the context of the work presented here towards fault-tolerant ECC, these elements correspond to ECSM modules as shown in

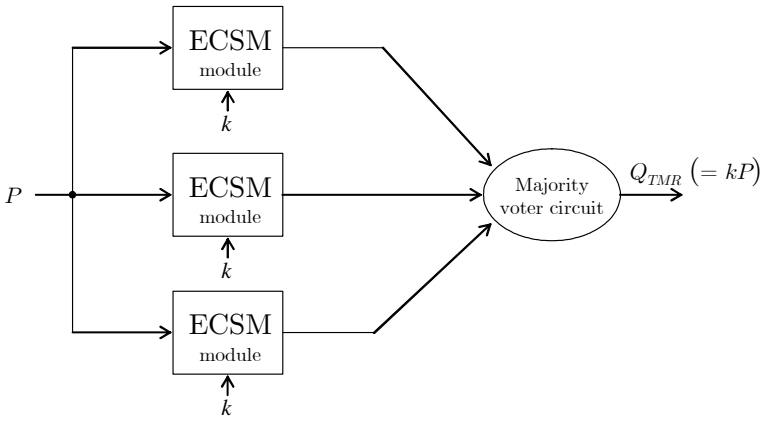

Figure 11: TMR based ECSM

Figure 11. The pair k and P are input to these three modules and all the outputs are connected to a majority voter circuit. Considering a perfect majority voter circuit, if errors occur in only one module then their effects can be masked and a correct output can be obtained.

A disadvantage of this TMR based ECSM is that it has considerable hardware overhead. It requires three modules and the majority voter circuit. This represents a significant increase in space and for constrained devices (e.g., smart cards) such a solution may not be attractive.

Let us assume that r_{ECSM} and r_V are the reliabilities of the ECSM module and the majority voter circuit, respectively. The reliability of the system, R_{TMR}, is related to the probability of two or three modules and the majority voter circuit produce the correct result, i.e.,

$$R_{TMR} = r_V(3r_{ECSM}^2 - 2r_{ECSM}^3).$$

Assuming that $r_V \approx 1$ we have

$$r_S \approx 3r_{ECSM}^2 - 2r_{ECSM}^3. \tag{7}$$

5.2 DMR based fault-tolerant ECSM

Extending the concept of DMR presented in Section 3, it is possible to have a fault-tolerant system as illustrated in Figure 12. Two ECSM modules operate in parallel using different inputs. One is computed normally with k and P as its inputs. For the other module, the negative of P (i.e., $-P$) and $(\#E) - k$ are used at the input. The point-on-the-curve checkers validate both computations and they control the output's selection with a multiplexor. The enable signal

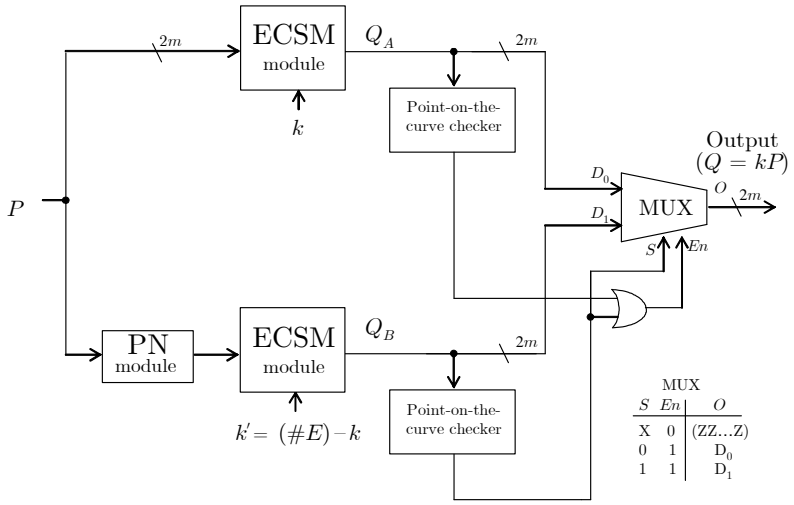

Figure 12: DMR based fault-tolerant ECSM

En is used for stopping the multiplexor inputs D_0 and D_1 to appear at the output in case D_0 and D_1 are not on the curve E.

This DMR based scheme can tolerate any faults provided that they are confined in only one of the two modules. Additionally, the scheme can detect situations where errors exist in both modules and hence it helps to avoid producing erroneous results at the output. It is more hardware efficient than the TMR based ECSM because it uses only two modules and it does not have the majority voter circuit.

Let r_{ECSM}, r_{PC}, r_{PN}, and r_{MUX} be the reliabilities of the ECSM module, the point-on-the-curve checker, the PN module, and the multiplexor respectively. The reliability of this system, R_{DMR}, is related to the probability that at least one of the two data paths and the multiplexor in Figure 12 work without errors. And, in the case that one data path fails, it is necessary that the point-on-the-curve checker return a 0 (i.e., $\widetilde{Q} \notin E$). The resulting expression for R_{DMR} is:

$R_{DMR} = r_{MUX}[r_{ECSM}^2\, r_{PC}^2\, r_{PN} + r_{ECSM}\, r_{PC}(1 - r_{ECSM}\, r_{PC}\, r_{PN})\, \Pr(\widetilde{Q} \notin E \,|\text{ECSM module failed}) + r_{ECSM}\, r_{PC} r_{PN}(1 - r_{ECSM}\, r_{PC})\, \Pr(\widetilde{Q} \notin E \,|\text{ECSM module failed})]$,

where $\Pr(\widetilde{Q} \notin E \,|\text{ECSM module failed})$ is the conditional probability of an erroneous result $\widetilde{Q}$ is not part of the original elliptic curve E given that the ECSM module has failed. If $r_{MUX} \approx 1$, $r_{PN} \approx 1$, and $r_{PC} \approx 1$, then we obtain:

$$R_{DMR} \approx r_{ECSM}\left(\frac{2^m-1}{2^{m-1}}\right) + r_{ECSM}^2\left(\frac{1-2^{m-1}}{2^{m-1}}\right). \quad (8)$$

5.3 Reliability comparison between TMR and DMR based fault-tolerant schemes

If we compare Equations (7) and (8) we observe that

$$r_{ECSM}\left(\frac{2^m-1}{2^{m-1}}\right) + r_{ECSM}^2\left(\frac{1-2^{m-1}}{2^{m-1}}\right) > 3r_{ECSM}^2 - 2r_{ECSM}^3,$$

for all $0 < r_{ECSM} < 1 - 2^{-m}$. For ECC used in practice we have $m \geq 163$, and the upper bound of this range of r_{ECSM} is close to unity.

Based on the above discussions, we can state that the reliability of the DMR based fault-tolerant structure is greater than that of TMR based system (see Figure 13). This is an interesting result because DMR requires less hardware than TMR system. The improved reliability of DMR is primarily due to the point-on-the-curve checker, which validates each computation and requires less area than an ECSM module. In the next section, we will compare the area requirements of these structures for a specific application.

6 Overhead cost

The error-detecting and fault-tolerant ECC structures presented in Section 3 and 5 require a number of extra modules, namely point-on-the-curve checker, PN module, register, comparator, majority voter circuit, and multiplexor. In this section we give costs of these components based on hardware implementation using FPGAs. We also give the time impact on ECSM operation, in terms of number of clock cycles, due to the inclusion of these extra components.

For ECSM operation, we use the performance results given in [14], where a NIST recommended elliptic curve over $GF(2^{163})$ has been used. The performance results are based on a Xilinx Virtex 2000E FPGA implementation and the ECSM operation uses a finite field multiplier, a squaring unit, and an adder. The area and timing results of these arithmetic units are:

Multiplier:	2364	slices and	4 cycles
Squaring unit:	165	slices and	1 cycle
Adder:	94	slices and	0 cycles

The entire module which performs ECSM requires 5009 slices and 17424 clock cycles for each scalar multiplication.

For implementing the extra modules (i.e., point-on-the-curve checker, PN module, register, comparator, majority voter circuit, and multiplexor), we have

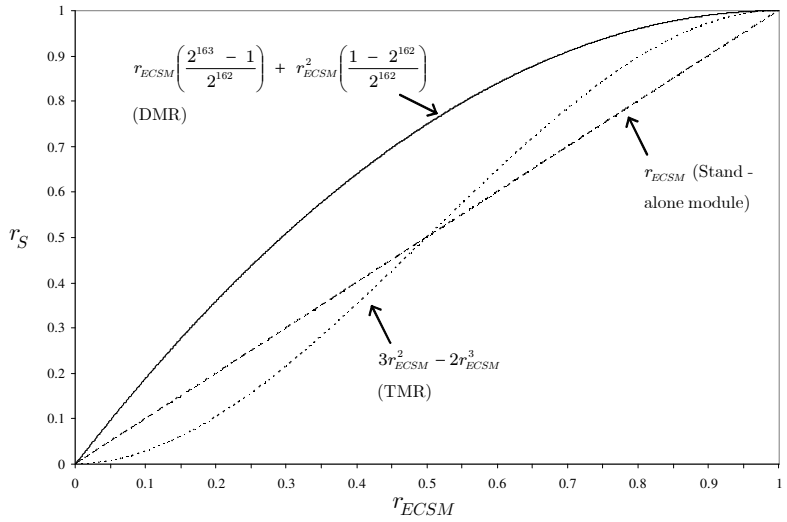

Figure 13: Reliability comparison between TMR and DMR based fault-tolerant schemes (m=163)

used the same FPGA to be consistent with the ECSM module of [14] . For the point-on-the-curve checker, in order to optimize the area requirement, we have used a low speed multiplier which occupies 286 slices and 163 cycles for each finite field multiplication. Additionally, this multiplier is used for the squaring operation to save space. The results of performance and cost for these extra modules are given in Table 7.

Element	**Slices**	**Cycles**
Point-on-the-curve checker	760	824
PN module	94	0
326-bit register	163	1
326-bit comparator	163	0
326-bit majority voter circuit	438	0
Multiplexor	163	0

Table 7: 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 8. The corresponding results for the fault-tolerant

System	Figure No.	Slices	Cycles
ECSM with point-on-the-curve checker	2	5769	18248
Time redundancy with the negative point	4	5523	34849
Time redundancy with variation on k and P	5	5429	34849
Time redundancy with a point-on-the-curve checker with variation on k and P	6	6189	35673
DMR with the negative point	7	10369	17424
DMR with variation on k and P	8	10275	17424
DMR with a point-on-the-curve checker with variation on k and P	9	11035	18248

Table 8: Performance and cost for error-checking systems over $GF(2^{163})$

structures of Section 5 are shown in Table 9.

System	Figure No.	Slices	Cycles
TMR based fault-tolerant ECSM	11	15465	17424
DMR based fault-tolerant ECSM	12	11795	18248

Table 9: Performance and cost for fault-tolerant systems over $GF(2^{163})$

7 Conclusions

In this report we have presented error-detecting and fault-tolerant structures for ECC. For the purpose of error detection, the concepts of point-on-the-curve checking, time redundancy, and hardware redundancy have been used. We have shown that the ECSM is a nearly self-checking operation using the point-on-the-curve checking process. However, in order to have a higher probability of error detection, this scheme can be used in combination with a time or hardware redundancy based scheme. For time redundancy schemes, we have proposed two types of encoding and decoding processes that are suitable for ECC. One is using the negative of the input P, and the other is based on the negative point and an alternative scalar ($k' = (\#E) - k$). Additionally, by generating single stuck-at faults exhaustively for a small ECSM prototype we have given experimental results that show the probability of undetected errors for the proposed error-detecting schemes. These results also show the benefit of combining hardware or time redundancy with the point on the curve checking process.

Traditionally, the concept of DMR has been associated with error-detecting systems only. However, for ECC we have shown that if one uses DMR with point-on-the-curve checking it is possible not only to detect but also correct some errors due by faults. If certain simple conditions are met, this DMR based fault-tolerant scheme is more efficient than the TMR based scheme.

A Example of undetected errors

Suppose we have the structure of Figure 4 implemented in hardware with one fault in the ECSM module. For simplicity assume that we have a stuck-at-1 fault, and it is located at the gate that gets bit i of the y-coordinate y_{p_i} as shown in Figure 14. Let the elliptic curve points that are input to the ECSM module at time t_0 and t_1 be $P = (x_p, y_p)$ and $-P = (x_{\overline{p}}, y_{\overline{p}})$ respectively, where $x_{\overline{p}} = x_p$ and $y_{\overline{p}} = x_p + y_p$. The input y-coordinates passing through the faulty gate are as follows:

$$\widetilde{y}_p = y_p \mid (00\ldots010...00),$$
$$\widetilde{y}_{\overline{p}} = (x_p + y_p) \mid (00\ldots010...00),$$

where the vector $(00\ldots010...00)$ has 0s in all its bits with the exception of bit position i, and the operator $\mid$ is a bit-wise OR.

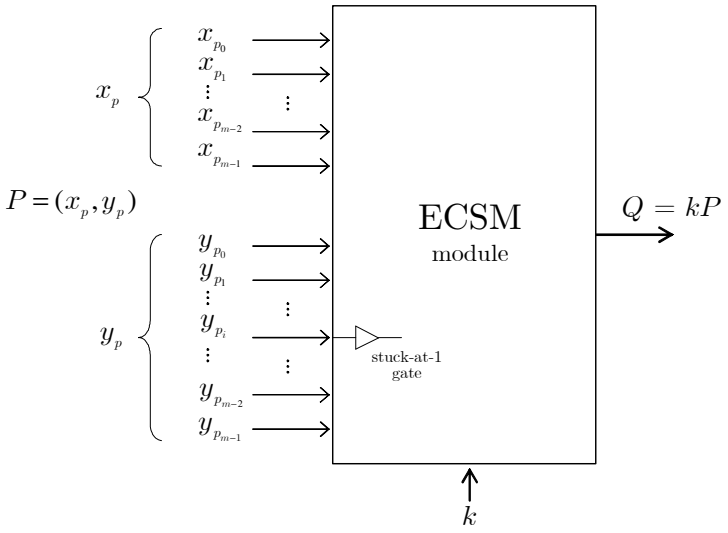

Figure 14: Stuck-at-1 fault at the gate that gets bit i of the y-coordinate

If $y_{p_i} = 1$ and $x_{p_i} + y_{p_i} = 1$, this fault does not produce any error in either of the two ECSMs (see Table 10). This is because the i-th bit of both y-coordinates is 1 and the fault does not change this bit's value. Any other combination of y_{p_i} and $x_{p_i} + y_{p_i}$ gives a different value as shown in Table 10. For the second (respectively third) case we can see that an original input point, P (respectively $-P$), is present at time t_0 (respectively t_1). For these two cases the results will be different. This is because it is necessary to have complementary points at both input modules in order to have the two outputs to match. For the fourth case, the input points are different from P and $-P$, say P_1 and P_2.

y_{p_i}	$x_{p_i} + y_{p_i}$	**ECSM input at t_0 (after the fault)**
1	1 ($x_{p_i} = 0$)	$(x_p, \widetilde{y}_p)$, where $\widetilde{y}_p = y_p$ (i.e., P)
1	0 ($x_{p_i} = 1$)	$(x_p, \widetilde{y}_p)$, where $\widetilde{y}_p = y_p$ (i.e., P)
0	1 ($x_{p_i} = 1$)	$(x_p, \widetilde{y}_p)$, where $\widetilde{y}_p \neq y_p$ (i.e., $\neq \pm P$)
0	0 ($x_{p_i} = 0$)	$(x_p, \widetilde{y}_p)$, where $\widetilde{y}_p \neq y_p$ (i.e., P_1)
y_{p_i}	$x_{p_i} + y_{p_i}$	**ECSM input at t_1 (after the fault)**
1	1 ($x_{p_i} = 0$)	$(x_{\overline{p}}, \widetilde{y}_{\overline{p}})$, where $\widetilde{y}_{\overline{p}} = y_{\overline{p}}$ (i.e., $-P$)
1	0 ($x_{p_i} = 1$)	$(x_{\overline{p}}, \widetilde{y}_{\overline{p}})$, where $\widetilde{y}_{\overline{p}} \neq y_{\overline{p}}$ (i.e., $\neq \pm P$)
0	1 ($x_{p_i} = 1$)	$(x_{\overline{p}}, \widetilde{y}_{\overline{p}})$, where $\widetilde{y}_{\overline{p}} = y_{\overline{p}}$ (i.e., $-P$)
0	0 ($x_{p_i} = 0$)	$(x_{\overline{p}}, \widetilde{y}_{\overline{p}})$, where $\widetilde{y}_{\overline{p}} \neq y_{\overline{p}}$ (i.e., P_2)

Table 10: Possible alteration of the input coordinates

The resultant finite field pairs are the negative of each other, say $P_1 = -P_2$. Furthermore, they are not part of the original elliptic curve $E(a, b)$, but they are part of another elliptic curve, say $\widetilde{E}(a, \widetilde{b})$. Because the point addition and doubling might not use the parameter b, the two ECSMs will be performed over $\widetilde{E}$. When the computations are complete, the two results will be the same but incorrect. In this particular case the system will fail by accepting an erroneous result. Similar analysis and results can be obtained for a stuck-at-0 fault at the same bit position of the y-coordinate (i.e., y_{p_i}), or if the fault is located at the gate that gets the i-th bit of the x-coordinate (i.e., x_{p_i}).

References

[1] 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] Biehl, I., Meyer, B., and Muller, V., "Differential Fault Attacks on Elliptic Curve Cryptosystems," Advances in Cryptology, Proc. CRYPTO 2000, LNCS 1880, pp. 131-146, Springer-Verlag, 2000.

[3] 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.

[4] Ciet, M., and Joye, M., "Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults," to appear in "Designs, Codes and Cryptography," Kluwer Academic, The Netherlands, 2004.

[5] Diffie, W., and Hellman, M. "New Directions in Cryptography," Proceedings of the AFIPS National Computer Conference, 1976.

[6] FIPS 186, Digital Signature Standard, Federal Information Processing Standards Publication 186, U.S. Department of Commerce/NIST, 1994.

[7] FIPS 186-2, Digital Signature Standard. Federal Information Processing Standards Publication 186-2, U.S. Department of Commerce/NIST, 2000.

[8] Hasan, M. A, "Look-up table-based large finite field multiplication in memory constrained cryptosystems," IEEE Transactions Computers, vol. 49, no. 7, pp. 749 - 758, July 2000.

[9] Hankerson D., Menezes, A., and Vanstone, S., "Guide to Elliptic Curve Cryptography," Springer, 2003.

[10] Itoh, T., and Tsujii, S., "Effective recursive algorithm for computing multiplicative inverses in $GF(2^m)$," Electronics Letters, Volume: 24, Issue: 6, pp. 334-335, March 1988.

[11] Johnson, B., "Fault-tolerant microprocessor-based systems," IEEE Micro, Vol 4, No 6, pp. 6-21, December 1984.

[12] 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.

[13] 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.

[14] Lutz, J., and Hasan, M. A., "High Performance Elliptic Curve Cryptographic Co-processor," Technical Report, CACR 2004-06, University of Waterloo, 2004.

[15] 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.

[16] 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.

[17] "The Reliability Design Handbook," Rome Air Development Center, Grifiss Air Force Base, N.Y., 1976.

[18] Standard Specifications for Public Key Cryptography, IEEE, P1363-2000.

[19] Von Neumann, J., "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Princeton University Press, pp. 43-98, 1956.

[20] Washington, L., "Elliptic Curves, Number Theory and Cryptography," CRC Press, 2003.

[21] 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.