# Design of Cryptographic Devices Resilient to Fault Injection Attacks Using Nonlinear Robust Codes

Kahraman D. Akdemir[1,3], Zhen Wang[2,3], Mark Karpovsky[2], and Berk Sunar[1]

[1] Department of Electrical and Computer Eng., Worcester Polytechnic Institute 100 Institute Road, Worcester, MA 01609 kahraman@wpi.edu, sunar@wpi.edu
[2] Reliable Computing Laboratory, Boston University 8 St Mary's Street, Boston, 02215 lark@bu.edu, markkar@bu.edu
[3] These authors contributed equally for the chapter.

## 1 Introduction

Active fault injection attacks pose a serious threat for many cryptographic applications, such as smartcards. Various countermeasures have been proposed to provide security against these attacks.

In [1, 2], a solution based on time redundancy by means of a double-data-rate (DDR) computation template was presented. Each computation is conducted twice and the results are compared to detect injected faults. Both clock edges were exploited to control the computation flow for the purpose of improving the throughput of the system. In [3, 4], the authors investigated the usage of dual-rail encoding for the protection of cryptographic devices against different types of side-channel attacks in asynchronous circuits.

The most commonly used fault detection technique is concurrent error detection (CED) which employ circuit level coding techniques, e.g. parity schemes, modular redundancy, etc. to produce and verify check digits after each computation. In [5], a secure AES architecture based on linear parity codes were proposed. The method could detect all errors of odd multiplicities with reasonable hardware overhead. In [6], an approach to fault tolerant public key cryptography based on redundant arithmetic in finite rings were presented. The method is closely related to cyclic binary and arithmetic codes. In [7], the authors proposed a CED technique that exploits the inverse relationships existing between encryption and decryption at various levels. A decryption is immediately conducted to verify the correctness of the encryption operation. A lightweight concurrent fault detection scheme for the S-box of AES was proposed in [8]. The structure of the S-box is divided into blocks and the predicted parities for these blocks are obtained and used for the fault detection.

Various fault attack countermeasures were compared in terms of the hardware overhead and the fault detection capabilities in [9].

Error detecting codes [10] are often used in cryptographic devices to detect errors caused by injected faults and prevent the leakage of useful information to attackers. Most of the proposed error detecting codes are linear codes like parity codes, Hamming codes and AN codes [7]. Protection architectures based on linear codes concentrate their error detecting abilities on errors with small multiplicities or errors of particular types, e.g. errors with odd multiplicities or byte errors. However, in the presence of unanticipated types of errors linear codes can provide little protection. Linear parity codes, for example, can detect no errors with even multiplicities.

It has never been proven nor argued that it is sufficient to only detect a particular subset of faults or errors to prevent a fault attack. The spectrum of available fault injection methods and the adaptive nature of an attacker suggests that it would be possible to bypass such protection by injecting a class of faults or errors which the cryptographic device has not been anticipating. Considering even only inexpensive non-invasive or semi-invasive fault attacks, there is a wide spectrum of the types of faults and injection methods an attacker has at his disposal [11].

Robust codes have been proposed as solution to the limitation of linear error detecting codes for detection of fault injection attacks [12]. These non-linear codes are designed to provide equal protection against all errors thereby eliminating possible weak areas in the protection that can be exploited by an attacker. Several variants of robust codes have been used to protect both private and public cryptographic algorithms. These variants allow several trade-offs in terms of robustness and hardware overhead for many architectures. *Robust* and *partially robust* codes have been used for the protection for both private [13, 12] and public key cryptosystems [14].

In this chapter we will present the basic constructions of robust codes and their applications for the design of secure cryptographic devices. As case studies, we discuss secure AES, secure ECC (Elliptic curve cryptography) and secure finite state machine (FSM) architectures based on robust codes, which are resilient to strong fault injection attacks. Sections 2, 3, 4 and 5 are contributed by the second and the third authors. Sections 6 and 7 are contributed by the first and the fourth authors. More details related to robust codes and their applications in the design of cryptographic devices can be found in [15, 13, 12, 14, 16, 6, 17, 18, 19].

## 2 Definition and Basic Properties of Robust Codes

We present all of the definitions in terms of binary codes. Most of the results can be easily generalized for codes over nonbinary fields.

**Definition 1.** ***(R-Robust Code)*** *A code $C \subseteq GF(2^n)$ is* ***R-robust*** *if the size of the intersection of the code $C$ and any of its translates $\tilde{C} = \{\tilde{x}|\tilde{x} = x + e, x \in C, e \in GF(2^n), e \neq 0\}$ is upper bounded by $R$:*

$$R = \max_{0 \neq e \in GF(2^n)} |\{x|x \in C, x + e \in C\}|. \tag{1}$$

*where $+$ is the componentwise addition modulo two.*

A graphic depiction of the definition of a robust code is shown in Figure 1. Let $C \subseteq GF(2^n)$, and $\tilde{C}_e$ be the set of all codewords of $C$ shifted by an element $e \in GF(2^n)$. The code C is *R-robust* if for any nonzero $e \in GF(2^n)$, the size of the intersection of C and $\tilde{C}_e$ is upperbounded by $R$.

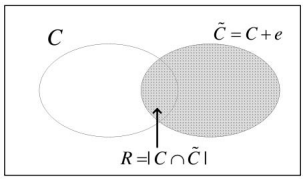



**Fig. 1.** Definition of robustness

The above defined robust codes have beneficial properties when the worst case error masking probability of the codes is considered. Let $M = |C|$ be the number of codewords in a code $C$. By definition of a $R$-robust code there are at most $R$ codewords which can mask any fixed error $e$. The error masking probability $Q(e)$ can be thus defined as

$$Q(e) = \frac{|\{x|x \in C, x + e \in C\}|}{M}. \tag{2}$$

Robust codes have no undetectable errors. For a $R$-robust code, the worst case probability of masking an error is at most $R/M$ for any error when the codewords of the robust code are assumed equiprobable. Clearly, robust codes which have a minimum $R$ for a given $M$ will also have the lowest probability of error masking and hence a predictable behavior in the presence of unpredictable error distributions since the worst case probability of masking any error is bounded. In the following sections we investigate the constructions and optimality of the codes followed by some examples of applications.

## 3 Bounds, Optimality, and Perfect Robust Codes

Based on the above definitions of the robust codes it is possible to derive the following main property for a R-robust code.

*Property 1.* If the code $C$ is R-robust then in the multiset $S_C = \{x_j + x_i | x_i, x_j \in C, x_i \neq x_j\}$, any element appears at most $R$ times.

Robust codes are optimal if they have the maximum number of codewords $M$ for a given $R$ and and length $n$. From Property 1, a relation on $R$, $n$ and $M$ of the code can be established.

$$M^2 - M \leq R(2^n - 1). \tag{3}$$

**Definition 2.** ***(Perfect Robust Code)*** *A R-robust code with n bits and M codewords satisfying $M^2 - M = R(2^n - 1)$ is* ***perfect****.*

Perfect robust codes are equivalent to classical combinatorial structures known as difference sets and symmetric designs [20]. It has been shown by Mann that all symmetric designs over binary fields and hence perfect binary robust codes exist only for even dimensions and are limited to the following parameters: $(2m+2, 2^{2m+1} \pm 2^m, 2^{2m} \pm 2^m)$ [21]. Moreover, *systematic* robust codes, which are often more practical for error detection in computer hardware due to their separation of data and check bits, cannot be perfect.

**Theorem 1.** *[22] For any systematic R-robust code with length n and k information bits, there are at least $2^{n-k}$ elements in $GF(2^n)$ which cannot be expressed as differences of two codewords.*

**Corollary 1.** *There are no perfect systematic robust codes.*

When perfect robust codes are not available, the best possible codes which maximize $M$ for a given $n$ and $R$ are referred to as *optimum* robust codes.

**Definition 3.** ***(Optimum Robust code)*** *Robust codes which have the maximum possible number of codewords M for a given length n and robustness R with respect to (3) are called* ***optimum****. For optimum codes adding any additional codewords would violate bound (3) and*

$$M^2 - M \leq R(2^n - 1) < M^2 + M. \tag{4}$$

*Example 1.* Consider the following binary code $C = \{000, 001, 010, 100\}$. It is easy to verify that for any nonzero element $e \in GF(2^3)$, there are at most two pairs of $c_1, c_2$ satisfying $e = c_1 + c_2$, where $+$ is the XOR operation. Hence the code is 2-robust.

The code is not perfect since equality does not hold for (3). The code, however is an optimum Robust code with $n = 3, M = 4, R = 2$. No other code can exist with the same $n$ and $R$ that has more codewords since 5 codewords would violate condition (3).

Several constructions of optimum systematic robust codes will be presented in the next section.

## 4 Constructions of Optimal Systematic Robust Codes

There is a strong relationship between robust codes, nonlinearity, and nonlinear functions since all robust codes are nonlinear. The parameters of systematic robust codes depend on nonlinearity of the encoding function of the codes.

We first review some basic definitions and properties of nonlinearity, a good survey of nonlinear functions can be found in [23].

Let $f$ be a function that maps elements from $GF(2^k)$ to $GF(2^r)$.

$$f : GF(2^k) \rightarrow GF(2^r) : a \rightarrow b = f(a). \tag{5}$$

The nonlinearity of the function can be measured by using *derivatives* $D_a f(x) = f(x + a) + f(x)$. Let

$$P_f = \max_{0 \neq a \in GF(2^k)} \max_{b \in GF(2^r)} Pr(D_a f(x) = b), \tag{6}$$

where $Pr(E)$ denotes the fraction of cases when $E$ occurred. The smaller the value of $P_f$, the higher the corresponding nonlinearity of $f$. For linear functions $P_f = 1$.

**Definition 4.** *A binary function $f : GF(2^k) \rightarrow GF(2^r)$ has **perfect nonlinearity** iff $P_f = \frac{1}{2^r}$.*

**Theorem 2.** *[24] Let $f$ be a nonlinear function that maps $GF(2^k)$ to $GF(2^r)$ where $k \geq r$, the set of vectors resulting from the concatenation of $x_1, x_2 : (x_1, x_2 = f(x_1))$ where $x_1 \in GF(2^k)$ and $x_2 \in GF(2^r)$ forms a robust systematic code with $R = 2^k P_f$, $n = k + r$ and $M = 2^k$.*

From Theorem 1, there are at least $2^{n-k}$ errors which will be detected with probability 1 by any systematic code with length $n$ and $k$ information bits. Thereby a more strict bound can be derived for systematic codes. In this case we have

$$M^2 - M \leq R(2^n - 2^{n-k}). \tag{7}$$

**Corollary 2.** *A systematic robust code*

$$C = \{(x_1, x_2 = f(x_1)) | x_1 \in GF(2^k), x_2 \in GF(2^r)\}$$

*is optimum if the encoding function $f$ is a perfect nonlinear function.*

*Remark 1.* The nonlinearity of the encoding function $f$ for systematic codes corresponds to the worst case error masking probability of the codes ([22, 24]). We have:

$$P_f = \max_{e=(e_1,e_2),e_1 \neq 0} Q(e) = \max_{e \in GF(2^{k+r})} Q(e). \tag{8}$$

where $e_1 \in GF(2^k), e_2 \in GF(2^r)$.

The following two constructions are examples of optimum robust codes based on perfect nonlinear functions.

**Construction 4.1** ***(Quadratic Systematic Code)*** *[24] Let* $x = (x_1, x_2, \cdots, x_{2s}, x_{2s+1})$, $x_i \in GF(2^r)$, $s \geq 1$. *A vector* $x \in GF(2^{(2s+1)r})$ *belongs to the code iff*

$$x_1 \bullet x_2 + x_3 \bullet x_4 + \cdots + x_{2s-1} \bullet x_{2s} = x_{2s+1}, \tag{9}$$

*where* $\bullet$ *is the multiplication in* $GF(2^r)$ *and* $\sum_{i=1}^{s} x_{2i-1} \bullet x_{2i}$ *is a perfect nonlinear function from* $GF(2^{2sr})$ *to* $GF(2^r)$. *The resulting code is a robust code with* $R = 2^{(2s-1)r}$, $n = (2s+1)r$ *and* $M = 2^{2sr}$. *The code is optimum with respect to Definition 3.*

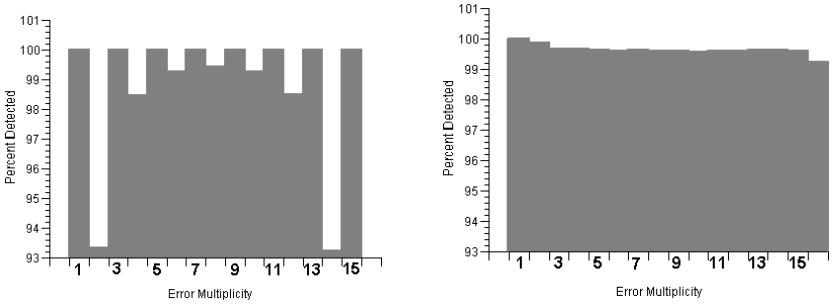
*Example 2.* **(Robust Parity)** Methods based on linear parity check codes are often used for on-line error detection in combinational circuits [25]. The linear 1-dim parity codes can detect all errors of odd multiplicities but offer no protection for errors of even multiplicities.

As an alternative to the 1-dim parity codes the quadratic systematic robust codes defined in Construction 4.1 can be used. When $r = 1$, the function defined in (9) is known as the bent function. The resulting systematic robust code has the same redundancy as the linear parity code. Unlike the linear parity code, the robust code will mask an error with a probability of at most $\frac{1}{2}$ regardless of the error multiplicity providing predictable error detection regardless of the error distribution.

Perfect nonlinear functions from $GF(2^k)$ to $GF(2^k)$ do not exist [23]. Functions with optimum nonlinearity in this case have $P_f = 2^{-k+1}$ and are called almost perfect nonlinear (APN) functions [26]. When $f$ are APN functions, the robust codes constructed as in Theorem 2 have $R = 2$. These codes are not optimum.

**Construction 4.2** ***(Robust Duplication Code)*** *Let* $x = (x_1, x_2), x_1, x_2 \in GF(2^r)(k = r)$ *. The robust duplication code* $C$ *contains all vectors* $x \in GF(2^{2r})$ *which satisfy* $x_1^3 = x_2$ *where all the computations are in* $GF(2^r)$. *The code is a 2-robust code with* $n = 2r$ *and* $M = 2^r$.

As an example, Figure 2 shows the percent of detectable errors as a function of error multiplicity (number of distorted bits) for 8-bit linear and robust duplication codes ($k = r = 8$). The detection capability of linear duplication codes depends largely on the multiplicity and type of the error. The scheme offers relatively poor protection for errors of even multiplicities, which can be exploited by the attacker to increase his chance of implementing a successful fault injection attack. On the contrary, robust duplication code has almost completely uniform error detection. This robust code has $R = 2$. Any error can be masked for at most two messages. Unlike for the linear codes, regardless of what subset of errors is chosen for this robust code the error masking probability is upper bounded by $2^{-7}$.




(a) (b)

**Fig. 2.** Percentages of errors detected versus error multiplicities for (a) 8-bit linear duplication (b) 8-bit robust duplication

### 4.1 Partially Robust Codes

Robust codes generally have higher complexity of encoding and decoding than classical linear codes. The quadratic systematic codes from Construction 4.1 require $s$ $r$-bit multipliers and $s-1$ $r$-bit componentwise additions. Assuming a $r$-bit multiplier requires $r^2$ two-input gates the encoder for the systematic quadratic code can be implemented with $sr^2 + r(s-1)$ 2-input gates.

As a tradeoff between robustness and the hardware overhead for computational devices, *partially robust codes* were introduced in [15]. These codes combine linear and nonlinear mappings to decrease the hardware overhead associated with generation of check bits by the predictor. The encoding of systematic partially robust code is performed first by using a linear function to compute the redundant $r$ check bits followed by nonlinear transformation. The use of the linear code as the first step in the encoding process typically results in a hardware savings in the encoder or predictor since the nonlinear function needs to only be computed based on the $r$ bit output of the linear block. The application of the nonlinear transformation reduces the number of undetectable errors thus increasing the robustness of the linear codes.

**Construction 4.3** ***(Partially Robust Codes)*** *[16] Let $f : GF(2^r) \rightarrow GF(2^r)$ be a nonlinear function with $P_f < 1$ and let $P : GF(2^k) \rightarrow GF(2^r)$, $r \leq k$ be a linear onto function. The set of words in the form $(x, f(P(x))$ form a code with $2^{k-r}$ undetectable errors.*

For partially robust codes described in Construction 4.3, the number of undetectable errors is reduced from $2^k$ to $2^{k-r}$ compared to the linear code with the same redundancy. Partially robust codes with $k = 128$ and $r = 32$ have been used in [13] for design of private key security devices on AES resistant to fault injection attacks. Implementation of this approach resulted in about 80% hardware overhead.

### 4.2 Minimum Distance Robust and Partially Robust Codes

A possible variant of the traditional robust robust codes is to include a minimum distance into the design criteria. Let $||e||$ denote the multiplicity of an error e (the number of ones in $e$). A robust code where the $Q(e) = 0$ for all errors with $||e|| < d$ is a *d-minimum distance robust code.*

Minimum distance robust codes are fully robust codes but also have a minimum distance larger than one. Since these codes are robust they have no undetectable errors and the worst case error masking probability is upper bounded and predictable. Moreover, they also provide for a guaranteed 100% probability of detection for errors with small multiplicities. Such codes can be useful for providing the highest protection against the most likely or most dangerous threat while maintaining a detection guarantee in case of an unexpected behaviour or attack. Moreover, minimum distance robust or partially robust codes with Hamming distance at least 3 can be used for not only error detection but also error correction [27]. For constructions of minimum distance robust and partially robust codes and applications of these codes in the design of secure cryptographic devices and reliable memory architectures, please refer to [16, 28, 29].

## 5 Secure AES Architectures Based on Nonlinear Codes

Encryption in AES-128 (AES with a 128-bit key) involves performing 10 rounds of transformations on a block of 128 bits with the last tenth round having one less transformation and with the first round being preceded by a round key addition. (The complete AES specification can be found in [30]) In each of the nine typical rounds there are four transformations: SBox, Shift Rows, Mix Columns, and Add Round Key. The last round differs from the rest in that it does not contain the Mix Columns transformation. The SBox transformation actually involves two operations: inversion in $GF(2^8)$ followed by an affine transform which involves a matrix multiplication over $GF(2)$, followed by addition of a constant vector. With the exception of inversion, all other transformations and operations are linear (Figure 3), i.e. they can be implemented using XOR gates only.

When considering only one round, the 128-bit data path can be divided into four identical independent 32-bit sections. Furthermore, in each of the four partitions the nonlinear inversion is performed on 8-bit data block. Thus, the nonlinear section is composed of 16 disjoint blocks and the linear portion composed of four identical disjoint blocks (Figure 4).

Based on this partitioning, redundant protection hardware can be designed for each of the two types of blocks. The details of each blocks method of protection are discussed in the next section.

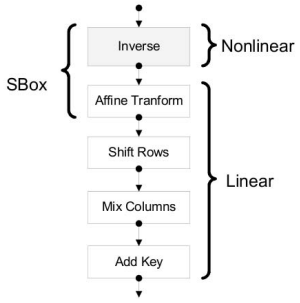



**Fig. 3.** Transformations involved in one typical round of encryption of AES

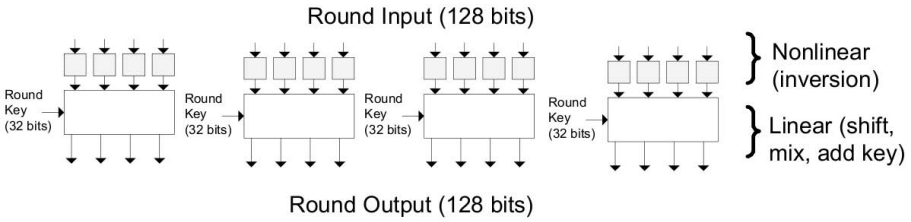



**Fig. 4.** The nonlinear portion of one round can be separated into 16 identical independent blocks. The linear portion can be separated into 4 identical independent blocks

### 5.1 Protection of the Nonlinear Block of AES

The nonlinear block performs inversion in $GF(2^8)$ . Since zero does not have an inverse, it is defined that the result of the inverse operation on zero is zero.

Let $x$ be the input to the nonlinear block. The fault detection circuitry for the nonlinear block can be based on multiplication in $GF(2^8)$ of input and output vectors to verify the condition

$$x \bullet x^{-1} = \begin{cases} 00000001 & \text{if } x \neq 0 \\ 00000000 & \text{if } x = 0. \end{cases}$$

*Remark 2.* $x^{-1}$ is an APN function over $GF(2^8)$ [26]. Hence the code $C$ defined by $\{(x, x^{-1})\}$ is a robust code with no undetectable errors.

To reduce the hardware overhead, we can compute the least $r < 8$ bits of the product instead of the whole 8-bit product. The probability that an error in the inverter will be missed is equal to the probability that two 8-bit vectors multiplied together will produce the expected $r$-bit constant $I_r$. When $x = 0$, $I_r = 0$. Otherwise $I_r$ has 1 for the least significant bit and 0 else where.

Let $e = (e_1, e_2)$ be the error vector, where $e_1 \in GF(2^8)$ is the error at the output of the original inverter and $e_2$ is the error at the output of the redundant portion. Then $e$ is missed iff

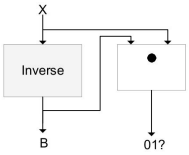



**Fig. 5.** Architecture for protection of nonlinear block. The redundant portion performs partial multiplication in $GF(2^8)$, $(r = 2)$.

$$((x^{-1} + e_1) \bullet x)_r = I_r + e_2, \tag{10}$$

i.e. the least significant $r$ bits of the product in $GF(2^8)$ is equal to $I_r + e_2$ or equivalently $x \bullet e_1 = e_2$.

For every nonzero $e = (e_1, e_2)$, there is at most one solution for $x$. Thereby all nonzero errors $e$ will be detected with a probability of at least $1 - 2^{-r}$. Moreover, since error detection depends on the input $x$, the probability that $e$ will be missed after $m$ random inputs is $2^{-rm}$.

In one round of encryption of AES there are $T = 16$ disjoint inverters, each with its own independent error detection. While for a single inverter the probability of missing an error is constant for all fault multiplicities that is not the case when multiple inverters are considered together. The probability that a fault will not be detected if it affects $t$ inverters is $q^t$ where $q$ is the probability of missing a fault in one inverter. (For the proposed architecture $q \leq 2^{-r}$)

Assuming that the distribution of faults is uniform, the probability that a fault of multiplicity $l$ will affect $t$ out of $T$ inverters can be determined as $P_T(t, l) = \frac{N_T(t,l)}{2^l}$, where $N_T(t, l) = \binom{T}{t}(t^l - \sum_{j=1}^{t-1} N_t(j, l))$. Thus, for AES and its $T = 16$ inverters the probability of missing a fault of multiplicity l in the whole nonlinear portion of encryption of one round is $Q_T(l) = \sum_{i=1}^{\min\{T,l\}} q^i P_T(i, l)$.

### 5.2 Protection of the Linear Block of AES

The general architecture used for protection with linear codes is presented in Figure 6. The architecture is composed of three major hardware components: original hardware, redundant hardware for predicting the $r$-bit signature $v$ (which is a linear combination of components of the output $x$ of the original device), and an error detecting network (EDN).

The signature predictor contains the majority of the redundant hardware. The $k$ bits of output of the original hardware and the $r$ redundant output bits of the signature predictor form the $n = k + r$ extended output of the device. The extended output forms a codeword of the systematic error-detecting code which can be used to detect errors in the original hardware or in the Predictor. It is the EDN which verifies, that the extended output of the device belongs to the corresponding code, if it does not then the EDN raises an error signal.

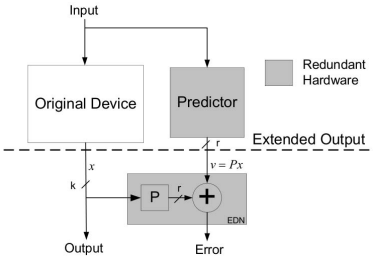



**Fig. 6.** General architecture for protection of hardware with linear error detecting codes

In a linear protection scheme the predicted $r$-bit signature $v$ of the Predictor is a linear combination of the $k$-bit output of the original device. ( $v = Px$ , where $P$ is a $r \times k$ check matrix for the linear code used for protection.)

With only a slight modification, the same architecture used for protection with linear codes can be used to provide protection based on the systematic nonlinear partially robust codes $\{x, (Px)^{-1}\}$. The transformation only requires an addition of two copies of one extra component for multiplicative inverse in $GF(2^r)$. The modified architecture is shown in Figure 7. The extended output of the device is now protected with the partially robust nonlinear code with the properties outlined above. An additional (and identical) multiplicative inverse is also needed in the EDN to verify the nonlinear signature. This transformation can be applied to any linear protection method regardless of what algorithm it is protecting.

As Table 1 shows, one very desirable consequence of the addition of inversion to create a robust code is the reduction in the number of undetectable errors. The number of undetectable errors is reduced from $2^k$ to $2^{k-r}$. When $k = r$, the code is robust and all nonzero errors are detectable.

Since large $r$ may be necessary to provide for a sufficiently high error-detecting probability the use of one large device which takes the multiplicative

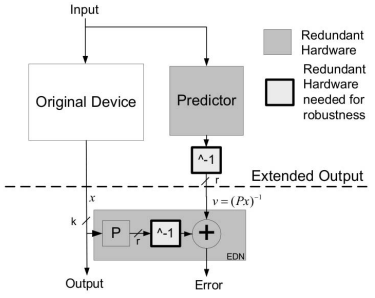



**Fig. 7.** Architecture for protection of hardware with nonlinear partially robust error detecting codes

inverse of all of the $r$-redundant bits might not be practical. Transforming an implementation protected by a linear code with $r = 32$ into a robust systematic code would require several thousands additional 2-input gates.

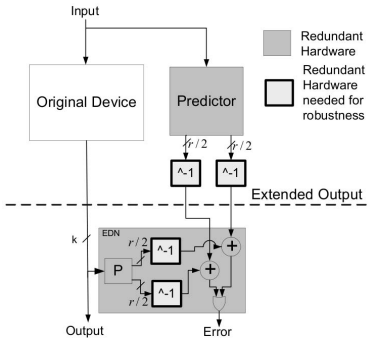



**Fig. 8.** Optimized architecture, the multiplicative inverse in split into $t = 2$ separate modules

It is possible to tradeoff the level of robustness for the amount of hardware overhead required to transform linear protection to protection based on systematic robust codes. Instead of taking one multiplicative inverse for all r-bit vectors, it is possible to divide the one large inversion into disjoint smaller inversions while retaining many of the robust properties outlined earlier. That is, we can replace multiplicative inverse in $GF(2^r)$ by $t$ $s$-bit disjoint inverses in $GF(2^s)$ to produce the nonlinear $r$ bit output ( $r = ts$ ). Thus, instead of having two $r$-bit multiplicative inverses in $GF(2^r)$ for the whole design, there could be $2t$ inverses in $GF(2^s)$ as it is presented in Figure 8 for $t = 2$. Since the number of two input gates to implement the inverse is proportional to the square of the number of bits at its input, a modification where $t = 2$ would result in roughly 50% decrease of an overhead associated with the architecture based on robust codes. As a consequence this also results in a slight decrease in the level of robustness and an in introduction of errors which are detected with different probabilities.

**Table 1.** Error detection capabilities of linear and nonlinear codes

| Error Detection Probability | Number of Errors | | |
|---|---|---|---|
| | linear | Robust ($r$ is odd) | Robust ($r$ is even) |
| 0 | $2^k$ | $2^{k-r}$ | $2^{k-r}$ |
| 1 | $2^n - 2^k$ | $2^{n-1} + 2^{k-1} - 2^{k-r}$ | $2^{n-1} + 2^{k-1} - 2^{k-r} + 2^k - 2^{k-r}$ |
| $1 - 2^{-r+1}$ | 0 | $2^{n-1} - 2^{k-1}$ | $2^{n-1} - 2^{k-1} - 2^{k+1} + 2^{k-r+1}$ |
| $1 - 2^{-r+2}$ | 0 | 0 | $2^k - 2^r$ |

**Protection of the Linear Block of AES Based on Minimum Distance Robust and Partially Robust Codes**

Protection architectures for the linear block of AES based on minimum distance robust and partially robust codes can be found in [16]. It was shown that for slow fault injection mechanism, where the attacker cannot change the injected faults at every clock cycle, minimum distance robust and partially robust codes can provide better fault detection capabilities than linear codes and traditional robust codes.

**Table 2.** Comparison of protection architectures of the linear block of AES based on different alternatives

| Code[1] | Number of 2-Input Gates | | Overhead(%) | $\|K_d\|$[2] | $Q_m$[3] |
|---|---|---|---|---|---|
| | Predictor | EDN | | | |
| Linear parity | 31 | 32 | 30% | $2^{32}$ | 0 |
| Robust parity | 185 | 32 | 100% | 0 | 0.5 |
| Min. dist. robust [16] | 196 | 64 | 120% | 0 | 0.5 |
| Hamming | 253 | 80 | 153% | $2^{32}$ | 0 |
| Gen. Vasil'ev [16] | 292 | 116 | 188% | $2^6$ | 0.5 |
| $(x, (Px)^3)$ [16] | 432 | 266 | 322% | $2^{26}$ | $2^{-5}$ |

[1]: All codes have 32 information bits.
[2]: $|K_d|$ is the number of undetectable errors.
[3]: $Q_m$ is the maximum error masking probability of detectable errors.

The hardware overhead in terms of the number of 2-input gates required for the implementation of the predictor and EDN for different alternatives are compared in Table 2. Compared to architectures based on linear codes, architectures based on robust or partially robust codes have better protection against strong fault injection attacks at the cost of higher hardware overhead. The architecture based on $(x, (Px)^3)$ partially robust codes requires more than 300% hardware overhead for the protection of linear block. Since the nonlinear block of AES is much larger than its linear block, the overall percentage hardware overhead of architectures based on robust or partially robust codes is much smaller than the data presented in Table 2. In [12], it was shown that architectures based on $(x, (Px)^3)$ partially robust codes require less than 80% overhead to protect the whole AES device. For more information about minimum distance robust codes and their applications, please refer to [16, 28, 29].

## 6 Secure FSM Design Based on Nonlinear Codes

Protecting the datapath of cryptographic algorithms is the focus of existing countermeasures against active fault attacks. However, even if the datapath

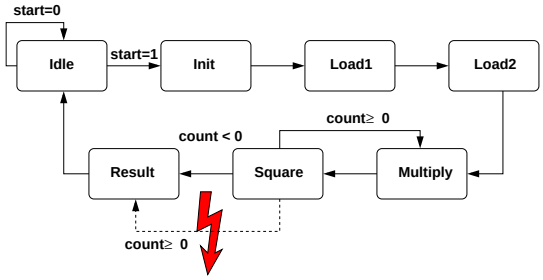



**Fig. 9.** Fault injection example on the control unit of the Montgomery Ladder Algorithm with Point of Attack indicated by the dashed transition [17]

is protected with the most secure scheme, the unprotected control unit may cause a serious vulnerability on the system. For instance, by injecting specifically chosen errors into the part of the IC that implements the control units, the adversary may bypass the encryption states in an FSM to conveniently gain access to secret information. Similarly, in a cryptographic authorization protocol the state which checks the validation of login information can be skipped with minimal effort. Thereby, the adversary can directly impersonate a valid user. Such attacks are indeed practical since states in an FSM are implemented using flip-flops, which can be easily attacked using bit-flips realized through fault injections. For example, Figure 9 shows an example attack scenario on the FSM of the Montgomery Ladder Algorithm [17]. In their paper, Gaubatz et al. show that the attacker can recover the secret exponent using this attack with mild effort. As a result, secure FSM design becomes an important problem.

### 6.1 The Error Detection Technique

In this section, we describe how to use the nonlinear error detection codes for securing the next-state logic of an FSM. However, we first provide the details of a specific nonlinear coding structure which is the main building block of our security scheme.

Initial version of the robust codes defined in [15] do not preserve arithmetic and hence cannot be applied to protect arithmetic structures against fault attacks. As a solution to this problem, a new type of non-linear arithmetic codes called "robust quadratic codes" is proposed in [14]. The following definition from [14] rigorously defines this particular class of binary codes.

**Definition 5.** *[14] Let C be an arithmetic single residue* $(n, k)$ *code with r=n-k redundant bits. Then* $C_p = \{(x, w) | x \in \mathbb{Z}_{2^k}, w = (x^2 \ mod \ p) \in GF(p) \}$ *where* $2^k - p < \epsilon$ *and r=k.*

In this definition, $\epsilon$ is used to represent a relatively small number. As will be explained in Theorem 3, $\epsilon$ determines how secure the coding scheme is.

As $\epsilon$ gets smaller, the utilized code becomes more secure. In addition, in this definition, $\mathbb{Z}_{2^k}$ is used to represent $k$-bit integers. While quantifying the error detection capability of $C_p$, [14] makes the following uniformity assumption:

**Assumption 1** *The values of $x \in \mathbb{Z}_{2^k}$ in $C_p$ are uniformly distributed and each value of $x$ is equal likely.*

Under this assumption, reference [14] provides the following bound on the error masking equation which quantifies the error detection performance of the nonlinear code $C_p$.

**Theorem 3.** *[14] For $C_p$, $max_{e\neq 0}(Q(e))=2^{-k}\cdot max(4, 2^k - p + 1)$.*

Our protection scheme is built on top of this coding structure. The main idea is to encode the variables of the FSM using the non-linear robust code $(n, k)$ as in Definition 5, and to use the error detection capability of this coding scheme for fault detection. However, a direct implementation of this coding scheme for an FSM would cause a serious security problem. Note that the specific error detection technique proposed by Gaubatz et al. works under the uniformity assumption which states that all the codewords are observed with the same probability (Assumption 1). This is a valid assumption if the code is applied to an arithmetic structure (such as an adder or a multiplier) where the inputs, and hence the outputs tend to be uniformly distributed. However, when the FSMs are concerned, this assumption becomes invalid because

1. depending on the FSM, some states may be visited more than others while the device is in operation, and
2. the number of inputs and states in an FSM are usually relatively small over a large domain.

Due to this non-uniform behavior, the security level provided by Theorem 3 does not apply if this non-linear coding scheme is directly applied to an FSM. In this case, the error detection probabilities of Theorem 3 will be much smaller because the number of valid codewords (fault free next-state values) determine the value of M=$|C_p|$ in the error masking probability of (2).

If the whole state space is being used uniformly by the valid next-state values (information carried by the code), then $|C_p|=2^k$ as in this theorem. However, in the non-uniform case of FSMs, the value of $|C_p|=t$, where $t$ is the number of states in the FSM. This alone dramatically increases the error masking probability of the detection scheme when $t \ll 2^k$ (which is usually the case for reasonable security levels). In addition, if there are some states that are visited more than others, this will decrease the effective value of $|C_p|$ even further, and hence will also cause an increase in the error masking probability.

Our solution to this problem is built around two innovative ideas:

- Arithmetic formulation of the next-state logic using Lagrange interpolation.

- Randomized embedding to solve the nonuniformity problem discussed before. In this method, each state value will have multiple images and this will provide unpredictability and uniformity.

More specifically, we first propose making state transitions work according to an arithmetic formula in a non-redundant field GF($q$), i.e. $s'$=$f(s,i)$ where $s'$ is the next-state value, $i$ is the input, and $s$ is the current-state value. In this setting, $s'$, $s$, and $i \in$ GF($q$).

However, this is not sufficient to solve the previously discussed non-uniformity problem because even though the state transitions are now working according to an arithmetic formula, the same non-uniform behavior is observed at the state registers. To solve this problem, we need unpredictability. In other words, the state machine variables must be practically unpredictable to the attacker. As a result, as the second step of the solution, we propose embedding the non-redundant field GF($q$) into a larger redundant ring $\mathbb{R}$=$\mathbb{Z}_M$ using a transformation $\phi$: GF($q$) $\mapsto \mathbb{R}$ (or $\phi$: $\mathbb{R} \mapsto \mathbb{R}$) where $\phi$ is a homomorphism. Therefore, all the arithmetic operations defined for GF($q$) can also be carried in $\mathbb{R}$. In our case, randomized embedding is achieved by defining the ring modulus $M$ and the scaling factor $S$ with $M$=$S \times q$ where $q$ is the prime defining the field GF($q$). In this setting, we define the function $\phi$ as $\phi(s)$=$s + R \times q \pmod M$ where $R$ is a randomly chosen value from $\mathbb{Z}_S$. This scaling effectively partitions the ring $\mathbb{R}$ into co-sets, of which only one contains the non-redundant codewords. As a result of this scaling, each state and input value now has $S$ images. In other words, the same state and input will now be represented by $S$ different values in the ring $\mathbb{R}$ depending on the value of the random $R$. This will increase the uniformity of the state values observed at the output of the next-state logic, and essentially increase the error detection capability of the non-linear coding technique we propose in this section. Note that when the state value is reduced using $q$, we obtain the non-redundant value of the state. We now formally define a randomized robust code by merging this embedding with the robust code definition introduced by Gaubatz et al. [14] in Definition 5 as follows.

**Definition 6.** *[18] We define the coset randomized robust code* $(n,k)$ *with* $r$=$n$-$k$ *redundant bits as* $C = \{(x,w) | x = y + R \times q \pmod M, \forall x$ *and* $\forall y \in \mathbb{Z}_M, w = x^2 \pmod p \in$ *GF(p)* $, R \in \mathbb{Z}_S\}$ *where* $r$=$k$, $k \geq max(\lceil \log_2 M \rceil, \lceil \log_2 p \rceil)$.

After the randomization of the next-state function $f$ is achieved, we can use individually robust arithmetic circuits such as multipliers, adders, etc. to build the proposed robust FSM. To achieve this, we utilize the non-linear code explained in Definition 6 to encode the state variables and inputs. Next, we use the individually robust arithmetic circuits that will work in the ring $\mathbb{R}$ to implement the next-state function $f$. Robust adder and multiplier implementation examples that work under the utilized code are provided in [14].

In the following theorem, we establish the error detection probability of our scheme. A detailed proof of this theorem can be found in [18].

**Theorem 4.** *[18] For the nonlinear code $C$ that is defined as in Definition 7, the error masking probability will be upper bounded by $S^{-1}{\cdot}max(4, 2^k - p + 1)$, where $S > max(4, 2^k - p + 1)$.*

*Example 3.* Consider the FSM of Figure 9. As discussed before, this FSM bounces between "Square" and "Multiply" states most of the time. If we select GF($q$)=GF(11) and $S$=390451572, then $M$=4294967292 and we will need a $(64, 32)$-code with $p$=4294967291. In this case, injected errors are detected with a probability of at least $1-S^{-1}{\cdot}\max(4, 2^k-p+1)=1-(2^{32}-4294967291+1)/(2 \times 390451572) \simeq 1 - 2^{-27}$. The denominator in this case becomes $2 \times S$ because the "Square" and "Multiply" states and their images will be uniformly distributed.

### 6.2 Case Study

In this section, we present the application of the proposed technique to the example FSM shown in Figure 9. The first step is to use two-level Lagrange interpolation to generate an algebraic polynomial for the next state logic. This polynomial represents the next-state $s'$ as a function of the input and current state variables $i$ and $s$, respectively. For the example FSM we investigate here, the following polynomial (see [18] for more details on the computation process) shows the result of the two-level Lagrange interpolation

$$s' = f(s,i) = (4i)s^6+(2+4i)s^5+(3+6i)s^4+(1+2i)s^3+(5+3i)s^2+(5+i)s+i. \tag{11}$$

This function can be implemented in an efficient way. The idea is to add time-redundancy and reuse the expensive hardware modules (e.g. the multiplier and the adder) in a serial manner. This can be achieved by rearranging the next-state polynomial of (11) using Horner's method. In this case, the next-state equation becomes

$$s' = ((((((4i)s+(2+4i))s+(3+6i))s+(1+2i))s+(5+3i))s+(5+i))s+i. \tag{12}$$

Once this polynomial is computed, the next step is to encode the input and current state values using the coset randomized code of Definition 6. In this case, the input will be $(i, |i^2|_p)$ and the current state will be $(s, |s^2|_p)$ where $i$, $s \in Z_M$. Next, using the computed function $f(s, i)$, we compute $(s', |(s')^2|_p)$. In this computation, all the main datapath operations are conducted over modulo $M$. Also note that each arithmetic unit (both in function $f$ and randomizer unit) is a robust one which implements nonlinear error detection individually. For example, all the multipliers in function $f$ and randomizer unit compute the output and its redundant checksum using the inputs and their redundant checksums. These individual components can detect an injected fault and raise an error signal. Next, using the randomizer unit, we compute $z=s' + R \times q \pmod M$. These operations are conducted using robust arithmetic units and are over modulo $M$ as well. The resulting randomized next

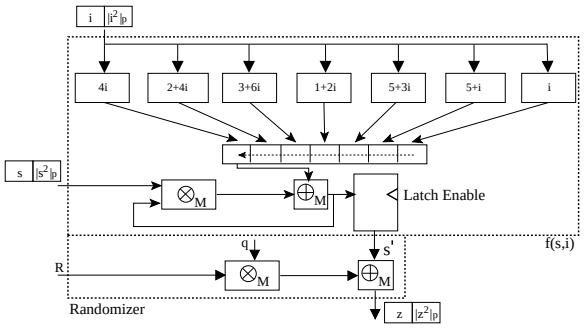



**Fig. 10.** Time redundant (serial) arithmetic hardware implementation of the next-state logic

state value is then fed as the current state value into the next state logic in the next iteration. Note that we do not need to recover $s'$ for the following next state computation. Since the function $f(s, i)$ and randomizer unit work modulo $M$, the next state value will always be a randomized image of the correct next state value. The non-redundant form of the next state value ($\in$ GF($q$)) can always be computed by reducing the randomized images modulo $q$. The resulting implementation is shown in Figure 10 for this specific example. For details on FSM security using nonlinear codes, the reader is referred to [18].

## 7 Secure ECC based on Nonlinear Codes

Elliptic Curve Cryptosystems (ECC) have also been a target of active fault attacks. In [31], Biehl et al. showed that using fault injection, ECC point multiplication can be forced to be computed over a less secure elliptic curve. As a result, it becomes relatively easy to solve the discrete log problem upon which the ECC is based on. They also proposed implementing bit faults during random moments of a multiplication operation and showed that it is possible to reveal the secret key $d$ in a bit-by-bit fashion. In [32], authors relaxed the assumptions of Biehl et al. in terms of the location and precision of the injected faults. Even with this new attacker model, their attack essentially recovers the (partial) secret in ECC discrete log problem. Similarly, in [33], Blomer et al. proposed the so-called "Sign Change Fault" attacks on the ECC based systems. Using this attack, they recover the secret scalar in polynomial time. In this section, we will discuss how nonlinear robust codes can be used to protect ECC operations.

### 7.1 Elliptic Curve Cryptography Overview

This section briefly describes the elliptic curve discrete logarithm problem (ECDLP) and ECC formulations over finite fields of prime character-

istics. A point $P$ of order $n$, selected over an elliptic curve $E$ defined over a finite field GF($p$), can be used to generate a cyclic subgroup $\langle P\rangle = \{\infty, P, 2P, 3P, \ldots, (\#n-1)P\}$ of $E(\mathrm{GF}(p))$.

The ECDLP is the underlying number theoretical problem used by ECC, and it is defined as determining the value $k \in [1, \#n-1]$, given a point $P \in E(\mathrm{GF}(p))$ of order $\#n$, and a point $Q = kP \in \langle P\rangle$. In a cryptosystem, the private key is obtained by selecting an integer $k$ randomly from the interval $[1, \#n-1]$. Then corresponding public key is the result of scalar point multiplication $Q = kP$, which is computed by a series of point additions and doublings.

ECC can be built upon two curves: 1) Weierstrass 2) Edwards. In this chapter, we focus on Edwards curves. However, note that the technique we propose is a generic one that can be applied to all elliptic curve structures (Weierstrass and Edwards) and all coordinate systems (projective and affine).

**Edwards Formulation for Elliptic Curves**

An elliptic curve $E$ defined over a prime field GF($p$) (with $p > 3$) can be written in the Edwards normal form as:

$$E(GF(p)) \,:\, x^2 + y^2 = c^2(1 + dx^2y^2), \tag{13}$$

where the parameter $c$ can be chosen as 1 without loss of generality. Addition of two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ in $E(\mathrm{GF}(p))$ resulting in a third point $P + Q = (x_3, y_3)$ in $E(\mathrm{GF}(p))$ can be computed as:

$$\begin{aligned} x_3 &= \frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2} \pmod p \\ y_3 &= \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \pmod p. \end{aligned} \tag{14}$$

This equation is valid even if $P = Q$, and it never results in point at infinity. An Edwards elliptic curve defined as in (13) is converted to homogeneous projective coordinates as $E(\mathrm{GF}(p)) \,:\, X^2 + Y^2{=}Z^4 + dX^2Y^2$ where $X = xZ$, $Y = yZ$. The following formulas compute the unified point addition and doubling (15), and optimized doubling (16) operations with projective coordinates [34]:

$$\begin{aligned} X_3 &= Z_1Z_2(X_1Y_2 + Y_1X_2)(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2) \pmod p \\ Y_3 &= Z_1Z_2(Y_1Y_2 - X_1X_2)(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2) \pmod p \\ Z_3 &= (Z_1^2Z_2^2 - dX_1X_2Y_1Y_2)(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2) \pmod p \end{aligned} \tag{15}$$

$$\begin{aligned} X_3 &= 2X_1Y_1(X_1^2 + Y_1^2 - 2Z_1^2) \pmod p \\ Y_3 &= (X_1^2 - Y_1^2)(X_1^2 + Y_1^2) \pmod p \\ Z_3 &= (X_1^2 + Y_1^2)(X_1^2 + Y_1^2 - 2Z_1^2) \pmod p. \end{aligned} \tag{16}$$

### 7.2 The Error Detection Technique

We mainly propose applying nonlinear codes to secure operations conducted over elliptic curves, i.e. point addition and doubling operations against active fault injection attacks. In this chapter, we are focusing on ECC structures based on prime fields GF($p$), yet a similar idea can be applied to protect elliptic curves that are defined over binary fields as well.

The main idea is to encode the coordinates of elliptic curve points using the systematic nonlinear $(n, k)$-code of Definition 5. This code essentially uses redundancy for error detection. We define the following error check function on a point coordinate $X \in \mathrm{GF}(p)$ to obtain a non-linear error check-sum

$$w = h(X) = X^2 \pmod{p} \in GF(p). \tag{17}$$

Consequently, the point coordinate $X$ is encoded as $(X, h(X))$. We now formally define a robust code by embedding the nonlinear code definition introduced by Gaubatz et al. [14] into elliptic curves as follows.

**Definition 7.** *[19] We define the prime field robust code $(n, k)$ with $r$=$n$-$k$ redundant bits as $C = \{(x, w)|x \in GF(p), w = x^2 \pmod{p} \in GF(p)\}$ where $r$=$k$, $k \geq \lceil \log_2 p \rceil$.*

In the non-redundant case, a point $P$ on an elliptic curve $E$ is represented as $P$=$(X, Y, Z)$ (assuming projective coordinates). However, with the new robust code definition we have, each point will be represented as $P$=$(X, X_w, Y, Y_w, Z, Z_w)$, where subscript $w$ is used to show the checksum portions.

The following theorem establishes the security level provided by the nonlinear code described in Definition 7 for any elliptic curve $E$. The detailed proof of this theorem can be found in [19].

**Theorem 5.** *[19] For the nonlinear code $C$ of Definition 7, the error masking probability is upper bounded by $max(4, 2^k - p + 1) \cdot (p + 1 - 2\sqrt{p})^{-1}$.*

*Example 4.* Consider the NIST recommended prime field curve P-192. For this curve, $(2^k - p + 1) = 18446744073709551618 \approx 2^{64}$. In this case, according to Hasse's theorem, the number of valid points on this curve will be at least $(p+1-2\sqrt{p}) \approx 2^{192}$. As a result, minimum error detection capability proposed by our scheme in this case will be $1 - 2^{64}/2^{192}$=$2^{-128}$.

### 7.3 Proposed Point Addition/Doubling Construction

In this section, we provide the secure implementation of the unified point addition/doubling operation for Edwards curves. This implementation utilizes the error detection technique that is described in Section 7.2. The main idea of the nonlinear error detection is to create two computation paths that are nonlinear to each other. As the first step to achieve this, the coordinates of

the input points in an operation (point addition or doubling) are encoded using the nonlinear code described in Definition 7. One of the nonlinear paths is the original non-redundant datapath. The second path, which is called the "predictor" block, runs in parallel to the non-redundant path, and essentially predicts the checksum of the results of the original computation. At this point, it is important to note that we do not simply replicate the original hardware to implement the predictor. For each datapath, the total operation count is expressed in terms of multiplications, divisions and addition&subtractions, where **M** stands for multiplication, **D** stands for division, and **A** stands for addition or subtraction.

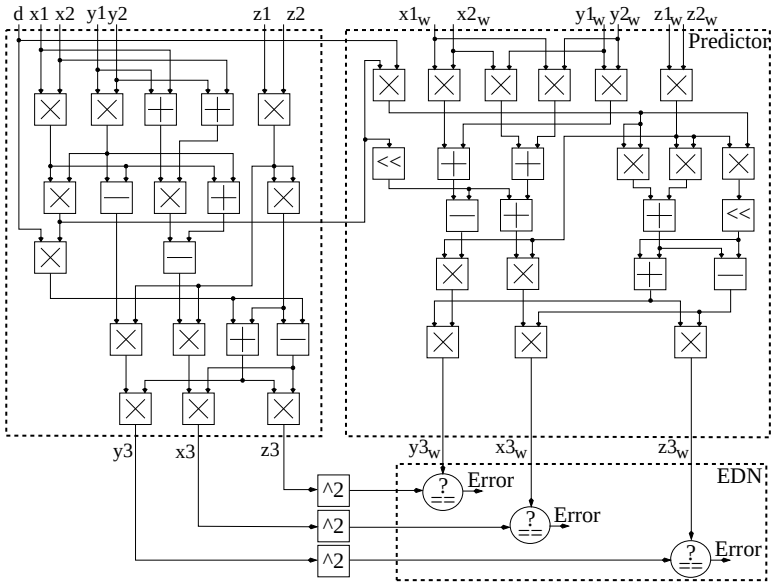



**Fig. 11.** Secure Edwards projective unified point addition

For Edwards curves that are using projective coordinates, the unified point addition operation computes the point $P3=(X_3, Y_3, Z_3)$ using the input points $P1=(X_1, Y_1, Z_1)$ and $P2=(X_2, Y_2, Z_2)$. The explicit formula that implements the point addition is shown in (15). In the following, we show how the predictor block works. It mainly computes the expected $X_{3w}$, $Y_{3w}$, $Z_{3w}$ using the inputs and their checksums. More specifically, the expected checksums should be

$$X_{3w} = (X_3)^2 = [Z_1 Z_2 (X_1 Y_2 + Y_1 X_2)(Z_1^2 Z_2^2 - dX_1 X_2 Y_1 Y_2)]^2,$$

$$Y_{3w} = (Y_3)^2 = [Z_1 Z_2 (Y_1 Y_2 - X_1 X_2)(Z_1^2 Z_2^2 + dX_1 X_2 Y_1 Y_2)]^2,$$

$$Z_{3w} = (Z_3)^2 = [(Z_1^2 Z_2^2 - dX_1 X_2 Y_1 Y_2)(Z_1^2 Z_2^2 + dX_1 X_2 Y_1 Y_2)]^2.$$

Next, we express the terms on the right hand side as a function of the inputs and their checksums. As an example, we show how to achieve this for $X_3$. The same method can also be applied to the $Y$ and $Z$ coordinates as well.

$$\begin{aligned} X_{3w} &= [Z_1 Z_2 (X_1 Y_2 + Y_1 X_2)(Z_1^2 Z_2^2 - dX_1 X_2 Y_1 Y_2)]^2 \\ &= Z_1^2 Z_2^2 (X_1^2 Y_2^2 + Y_1^2 X_2^2 + 2X_1 Y_2 Y_1 X_2)(Z_1^4 Z_2^4 - 2Z_1^2 Z_2^2 dX_1 X_2 Y_1 Y_2 + \\ &\quad d^2 X_1^2 X_2^2 Y_1^2 Y_2^2) \\ &= Z_{1w} Z_{2w} (X_{1w} Y_{2w} + Y_{1w} X_{2w} + 2\alpha)(Z_{1w}^2 Z_{2w}^2 - 2Z_{1w} Z_{2w} d\alpha + \\ &\quad d^2 X_{1w} X_{2w} Y_{1w} Y_{2w}), \end{aligned}$$

where $\alpha$=$X_1 X_2 Y_1 Y_2$. After some algebra, we get the following equation array for each coordinate of the resulting point $P3$. These equations mainly represent the function implemented by the predictor unit in our design.

$$\begin{array}{llll} \alpha = X_1 X_2 Y_1 Y_2; & & & \\ A = Z_{1w} Z_{2w}; & B = X_{1w} Y_{2w}; & C = Y_{1w} X_{2w}; & D = X_{1w} X_{2w}; \\ E = Y_{1w} Y_{2w}; & F = d\alpha; & G = B + C + 2\alpha; & H = D + E - 2\alpha; \\ K = A^2 + F^2; & L = AF; & M = K - 2L; & N = K + 2L; \end{array}$$

$$X_{3w} = AGM; \quad Y_{3w} = AHN; \; Z_{3w} = MN;$$

The total operation count for this predictor unit will be $14\mathbf{M} + 7\mathbf{A}$. Note that, all the operations in this setup are modulo $p$, where $p$ is the prime that generates the finite field the elliptic curve is defined over.

The hardware implementation of this technique is shown in Figure 11. In this figure, the block on the left is the original, non-redundant datapath that computes the unified point addition. The predictor block mainly implements the $X_{3w}$, $Y_{3w}$, and $Z_{3w}$ computations defined above. Next, the output coordinates are squared to compute their checksums. Finally, the error detection network (EDN) compares the results of these two paths. If all the resuts match, this means that the conducted operation is fault free. However, if there is a mismatch in any one of the coordinate comparisons, this points to an injected fault. Hence, an error signal is asserted. Once the error signal is asserted, either the secret can be flushed or the device can be reset. For details on ECC security using nonlinear codes, the reader is referred to [19].

## 8 Conclusion

Nonlinear robust codes is one of the most effective solutions against active fault injection attacks. In this chapter, we provided a profound analysis of nonlinear codes by investigating various constructions and their applications. More specifically, we discussed protection of the AES datapath, FSMs, and ECCs using nonlinear robust codes.

## References


1. P. Maistri and R. Leveugle, "Double-Data-Rate Computation as a Countermeasure against Fault Analysis," *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1528–1539, 2008.
2. P. Maistri, P. Vanhauwaert, and R. Leveugle, "Evaluation of Register-Level Protection Techniques for the Advanced Encryption Standard by Multi-Level Fault Injections," in *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07*, 2007, pp. 499–507.
3. P. Cunningham, R. Anderson, R. Mullins, G. Taylor, and S. Moore, "Improving Smart Card Security Using Self-Timed Circuits," in *Proceedings of the 8th International Symposium on Asynchronus Circuits and Systems*. IEEE Computer Society Washington, DC, USA, 2002.
4. K. Kulikowski, V. Venkataraman, Z. Wang, A. Taubin, and M. Karpovsky, "Asynchronous balanced gates tolerant to interconnect variability," in *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008*, 2008, pp. 3190–3193.
5. G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 492–505, 2003.
6. G. Gaubatz and B. Sunar, "Robust finite field arithmetic for fault-tolerant public-key cryptography," in *2nd Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2005*, L. Breveglieri and I. Koren, Eds., September 2005.
7. R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 21, no. 12, pp. 1509–1517, 2002.
8. M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight concurrent fault detection scheme for the aes s-boxes using normal basis," in *Cryptographic Hardware and Embedded Systems CHES 2008*.
9. T. G. Malkin, F.-X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2006.
10. F.J.MacWilliams and N.J.Sloane, *The Theory of Error-Correcting Codes*, 1977.
11. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *In Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006.
12. M. Karpovsky, K. Kulikowski, and A.Taubin, "Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard," ser. Proc. Int. Conference on Dependable Systems and Networks (DNS 2004), July 2004.
13. ——, "Differential fault analysis attack resistant architectures for the advanced encryption standard," ser. Proc. IFIP World Computing Congress, Cardis, Aug 2004, pp. 177–193.
14. G. Gaubatz, B. Sunar, and M. Karpovsky, "Non-linear residue codes for robust public-key arithmetic," in *Proceedings of the 3rd Workshop on Fault Tolerance and Diagnosis in Cryptography (FTDC)*. Springer, 2006, pp. 173–184.

15. M. G. Karpovsky and A. Taubin, "New class of nonlinear systematic error detecting codes," *IEEE Trans. on Information Theory*, vol. 50, no. 8, pp. 1818–1820, 2004.
16. K. J. Kulikowski, Z. Wang, and M. G. Karpovsky, "Comparative analysis of robust fault attack resistant architectures for public and private cryptosystems," in *FDTC '08: Proceedings of the 2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 41–50.
17. G. Gaubatz, B. Sunar, and E. Savas, "Sequential circuit design for embedded cryptographic applications resilient to adversarial faults," *IEEE Transactions on Computers*, vol. 57, no. 1, pp. 126–138, 2008.
18. K. D. Akdemir and B. Sunar, "A generic approach for hardening state machines against strong adversaries," *IET Computers and Digital Techniques (Accepted for Publication)*.
19. K. D. Akdemir, D. Karakoyunlu, and B. Sunar, "Nonlinear error detection for elliptic curve cryptosystems," *IET Information Security (Under Review)*.
20. D. Jungnickel and A. Pott, *Difference sets, sequences and their correlation properties*. Kluwer Academic Publishers, 1999, ch. Difference sets: An introduction.
21. T. Beth, D. Jungnickel, and H. Lenz, *Design Theory, Volume 1*. Cambridge University Press, 1999.
22. M. Karpovsky, K. Kulikowski, and Z. Wang, "On-line self error detection with equal protection against all errors," *Int. Journal of Highly Reliable Electronic System Design*, 2008.
23. C. Carlet and C. Ding, "Highly nonlinear mappings," *Journal of Complexity*, vol. 20, no. 2-3, 2004.
24. M.G.Karpovsky, K. Kulikowski, and W. Zhen, "Robust error detection in communication and computation channels," in *Keynote paper, Int. Workshop on Spectral Techniques*, 2007.
25. P. K. Lala, *Self-Checking and Fault-Tolerant Digital Design*. Morgan Kaufman, 2001.
26. M. S. Maxwell, "Almost perfect nonlinear functions and related combinatorial structures," Ph.D. dissertation, ISU, 2005.
27. K. Kulikowski, "Codes and circuits for secure hardware design," Ph.D. dissertation, Boston University, 2009.
28. Z. Wang, M. Karpovsky, and K. Kulikowski, "Replacing linear hamming codes by robust nonlinear codes results in a reliability improvement of memories," in *Dependable Systems and Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, 29 2009-July 2 2009, pp. 514–523.
29. Z. Wang, M. Karpovsky, and A. Joshi, "Reliable MLC NAND flash memories based on nonlinear t-error-correcting codes," in *Dependable Systems and Networks, 2010. DSN '10. IEEE/IFIP International Conference on*, 2010.
30. "Fips pub 197: Advanced encryption standard," http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
31. I. Biehl, B. Meyer, and V. Muller, "Differential fault attacks on elliptic curve cryptosystems," *Lecture Notes in Computer Science*, pp. 131–146, 2000.
32. M. Ciet and M. Joye, "Elliptic curve cryptosystems in the presence of permanent and transient faults," *Designs, Codes and Cryptography*, vol. 36, no. 1, pp. 33–43, 2005.

33. J. Blomer, M. Otto, and J. Seifert, "Sign change fault attacks on elliptic curve cryptosystems," in *Proceedings of the 2nd Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2005)*. Springer, 2005, pp. 25–40.
34. D. Bernstein and T. Lange, "Faster Addition and Doubling on Elliptic Curves," *ASIACRYPT 2007, Kuching, Malaysia*, vol. 4833, pp. 29–50, 2007.