Robust Codes and Robust, Fault Tolerant Architectures of the Advanced Encryption Standard

Konrad J. Kulikowski, Mark G. Karpovsky, Alexander Taubin
Reliable Computing Laboratory, Boston University
8 Saint Mary's Street, Boston, MA 02215
{konkul, markkar, taubin}@bu.edu

Abstract—
Hardware implementations of cryptographic algorithms are vulnerable to fault analysis attacks. Methods based on traditional fault-tolerant architectures are not suited for protection against these attacks. To detect these attacks we propose an architecture based on Robust nonlinear systematic error-detecting codes. These nonlinear codes are capable of providing uniform error detecting coverage independently of the error distributions. They make no assumptions about what faults or errors will be injected by an attacker. Architectures based on these Robust construction have fewer undetectable errors than linear codes with the same *(n,k)*. We present the general properties and construction methods of these codes as well as their application for the protection of a cryptographic devices implementing the Advanced Encryption Standard.

Keywords: side-channel attacks, Robust codes, fault attacks, Advanced Encryption Standard

I. INTRODUCTION

Hardware implementations of cryptographic algorithms are vulnerable to malicious analyses that exploit the physical properties of the designs. Power consumption, electro-magnetic radiation, execution time, and behavior in the presence of faults can all be used to drastically decrease the complexity of cryptanalysis. Mobile cryptographic devices such as smartcards and mobile computers are especially vulnerable since the physical hardware implementing the algorithms, and hence the side-channel information, is easily accessible.

The side-channel attacks of interest to this paper are Differential Fault Analysis (DFA) attacks. DFA attacks use information obtained from an incorrectly functioning implementation of an algorithm to derive the secret information. DFA attacks were first proposed by Biham et al. [1] against hardware implementations of the Data Encryption Standard (DES). They have since been extended to other symmetric key algorithms, such as the Advanced Encryption Standard (AES) in [3-6]. Incorrect operation can result from faults within the circuit (permanent or transient) which may be due to natural effects or be maliciously induced. DFA attacks pose a serious security threat since faults and fault attacks can be actively and adaptively injected in an effort to bypass a protection mechanism [15].

Existing techniques proposed for protection against the DFA attacks are based methods adapted from traditional architectures used for reliability purposes in non cryptographic hardware. However, there are fundamental limitations and potential problems of reusing traditional fault-tolerant designs based on error-detecting codes due to the differences in the fault and error models for computation channels and those which should be considered for a cryptographic device subject to an active fault attack. In traditional fault-tolerant architectures the protection is tailored for a specific error model obtained by analyzing the typical natural channel properties and error rates. As a result of the expected low error rates and fault multiplicities, traditional fault-tolerant architectures are concerned mostly with detection or correction of faults and errors of a very specific class resulting in protection based on codes with a small minimum distance (i.e. parity, Hamming, Triple Modular Redundancy (TMR)).

For cryptographic devices an attacker has the capability of adjusting the fault injection mechanisms and techniques to inject faults and errors of almost any multiplicity and type. Predicting the typical error distributions or even more importantly assuming only faults of a low multiplicity for an attack are not appropriate. Typically most "secure" systems fail because of the untypical behavior of an adversary which was not predicted and planned for. Therefore, we argue that traditional error-detecting approaches which assume a fault or error model and distribution are not ideally suited for the protection of cryptographic devices in face of an active adversary. We therefore propose modifications of traditional methods which aim at providing *uniform protection against all errors without (or which minimize) any assumptions on the error and fault distributions, capabilities and methods of an attacker.* Specifically we target three main criteria for evaluating the effectiveness and practicality of a protection scheme for fault attacks:

1. The number of undetectable output distortions or output errors
2. The maximum probability of missing an error or class of errors
3. Spatial and temporal overhead

The protection method proposed in this paper minimizes the size and weakness of the least protected areas under given limitations imposed by overheads while minimizing the hardware overhead associated with the decoding network. We propose protection methods based on a class of *new nonlinear systematic error-detecting codes* called Robust codes. These nonlinear codes are robust in terms of equal protection against all errors. Robust nonsystematic codes which provide equal probabilities of detection for all error distributions were presented in [8-9]. The methods in this paper are based on systematic robust nonlinear error detecting codes which are similar to those described in [10]. We start with an introduction of the codes and their basic properties which can be useful for hardware implementations. Next we emphasize some of the benefits of the proposed constructions and finish with some practical case study of the benefits of using Robust constructions to redistribute the error detecting power in a hardware implementation of the Advanced Encryption Standard secure against the DFA attacks.

II. SYSTEMATIC ROBUST CODES

Let $C \subseteq GF(q^n)$ be a (n, M) -code, where $M = |C|$

Definition 1

The code *C* is *Robust* with respect to its error-masking probability iff the probability $Q(e)$ of missing an error *e* is less than one for all nonzero errors *e*:

$$Q(e) = \frac{|\{w \mid w \in C, w+e \in C\}|}{|C|} < 1, \ e \neq 0 \tag{1.1}$$

where $w, e \in GF(q^n)$.

Definition 2

The code *C* is *uniformly Robust* with respect to its error-masking probability iff the probability $Q(e)$ is constant independently of the nonzero error *e*:

$$Q(e) = \frac{|\{w \mid w \in C, w+e \in C\}|}{|C|} = const., \ e \neq 0, const < 1 \tag{1.2}$$

Definition 3

A Robust code where $R = \max |\{w \mid w \in C, w+e \in C\}|$ for all errors $e \neq 0$ is called a *R-Robust code* and is denoted by C_R.

A graphic depiction of the definitions of the properties of a Robust code is shown in Figure 1. Let C be the set of all codewords of length *n*, and let $\tilde{C}$ be the set of all codewords of set C shifted by an element $e \in GF(q^n)$, $\tilde{C} = \{\tilde{w} \mid \tilde{w} = w + e, w \in C\}$, then the code C is Robust if for any $e \in GF(q^n)$, $e \neq 0$ the intersection of the two sets C and $\tilde{C}$ is less than the size of the code: $|C| > |C \cap \tilde{C}|$. Additionally, if for any *e*, the size of the intersection is less than or equal to $R < |C|$, then the code is a R-Robust code. If $R < |C|$ is constant for all shifts of the code, then the code C is said to be uniformly Robust.

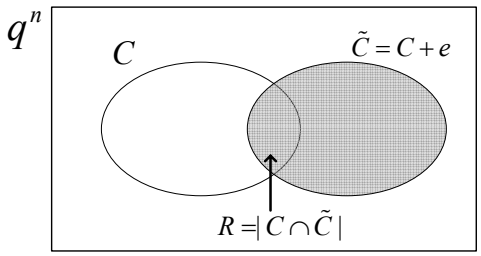

Figure 1. Definition of Robustness

One construction of systematic Robust error detecting codes is based on the use perfect nonlinear functions [14]. We now provide a definition of these functions and the corresponding Robust code construction.

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

$$f : GF(q^k) \to GF(q^r) : a \to b = f(a)$$

The nonlinearity of the function can be measured by using derivatives $D_a f(x) = f(x+a) - f(x)$. The nonlinearity measure can be defined by (from [14])

$$P_f = \max_{0 \neq a \in GF(q^k)} \max_{b \in GF(q^r)} \Pr(D_a f(x) = b) \tag{1.3}$$

where $\Pr(E)$ denotes the probability of occurrence of event E. The smaller the value of P_f, the higher the corresponding nonlinearity of f. For linear functions $P_f = 1$.

Definition 4

A function $f : GF(q^k) \to GF(q^r)$ has *perfect nonlinearity* if $P_f = \frac{1}{q^r}$.

Several constructions and an overview of perfect nonlinear functions can be found in [14].

Theorem 1

Let f be a perfect nonlinear function that maps $GF(q^k)$ to $GF(q^r)$ where $k \geq r$, the set of vectors resulting from the concatenation of $a, b : \{(a, b = f(a))\}$ where $a \in GF(q^k)$ and $b \in GF(q^r)$ forms a uniform (q^{k-r})-Robust systematic error-detecting code.

Proof:

The error $e = (e_x, e_y)$ ($e_x \in GF(q^k)$ $e_y \in GF(q^r)$) will be masked iff $f(x + e_x) - f(x) = e_y$, $x \in GF(q^k)$ which is exactly when $D_{e_x} f(x) = e_y$. By the definition of a perfect nonlinear function (Definition 4) we have that $P_f = \max_{0 \neq a \in GF(2^k)} \max_{b \in GF(2^r)} \Pr(D_a f(x) = b) = \frac{1}{q^r}$. Since there are q^k vectors in the code and only q^r solutions of the above error-masking equation, all errors are detectable and hence the codes are uniformly (q^{k-r})-Robust. □

Example 1

Let $w \in GF(q^k)$ where k is even. The following function f from ($GF(q^k)$,+) to $(GF(q), +)$

$$f(w=(x_1x_2...x_k))=x_1x_2+x_3x_4+...+x_{k-1}x_k$$

has perfect nonlinearity $P_f=\frac{1}{q}$ where $x_i\in GF(q^k)$. The set of words resulting from the concatenation of w and $f(w)$, where $w\in GF(q^k)$ and $f(w)\in GF(q)$ forms a uniformly q^{k-1}-Robust error-detecting code:

$$C_{q^{k-1}}=\{(w=(x_1x_2...x_k),f(w))\mid w\in GF(q^k),k\ is\ even,x_i\in GF(q),f(v)=x_1x_2+...+x_{k-1}x_k\}$$

To see that the resulting code is Robust consider the error masking condition for a fixed error $e=(e_1e_2...e_ke_{k+1})$ where $e_i\in GF(q)$. An error $e\neq 0$ is missed for an intended codeword $w=(x_1x_2...x_kx_{k-1})$, $w\in C_{q^{k-1}}$ iff

$$(x_1+e_1)(x_2+e_2)+...+(x_{k-1}+e_{k-1})(x_k+e_k)=x_{k+1}+e_{k+1}.$$

Since by the definition of the code $x_{k+1}=x_1x_2+...+x_{k-1}x_k$ the error masking equation simplifies to

$$x_1e_2+x_2e_1+...+x_{k-1}e_k+x_ke_{k-1}+e_1e_2+...+e_{k-1}e_k+e_{k+1}=0.$$

Analyzing equation it is easy to see that each the equation is satisfied and the error $e\neq 0$ is masked for exactly q^{k-1} messages. □

For hardware implementation and protection the *binary* Robust codes are the most important and practical. From this point all discussion will assume a binary Robust code but almost all of the properties and results can be generalized to the q-ary (q is prime) case.

Robust codes have several properties which can be used to minimize or simplify code construction. We will present below three important properties of these codes. The proofs of the properties follow directly from Definitions 1-3

Property 1

If the code $C_R=\{w_1,w_2,...,w_k\}$ $w_i\in GF(2^n)$ is R-Robust then R is even and there are $R/2$ non intersecting pairs w_i+w_j which have the same sum, $w_i+w_j=\Delta_k$.

Property 2

If the code $C_R=\{w_1,w_2,...,w_k\}$ $w_i\in GF(2^n)$ is a R-Robust code so is the shifted code

$C_R'=\{w_1\oplus e,w_2\oplus e,...,w_k\oplus e\}$ $w_i,e\in GF(2^n)$ for any shift e.

Property 3

If the code $C_R=\{w_1,w_2,...,w_k\}$ $w_i\in GF(2^n)$ is a R-Robust code so is

$C_R'=\{w_1e,w_2e,...,w_ke\}$ $w_i,e\in GF(2^n),e\neq 0$ where multiplication is in the field *GF(2ⁿ)*.

While nonlinearity is a necessary condition for Robustness, there is direct relationship between good Robust codes and good linear codes. Good 2-Robust codes can be used to construct good double error correcting linear codes and good double-error correcting linear codes can be used to construct good 2-Robust error-detecting codes.

Theorem 2

Let V be a double error correcting linear (n,k) code with a $((r=n-k)\times n)$ parity check matrix **H**. The columns of the check matrix **H** form the codewords of an *r*-bit 2-Robust code C_2 where $|C_2|=n$.

Proof:

Let $H=[h_0,h_1,...,h_{n-1}]$ be the parity-check matrix of a linear code V with minimum distance d. By definition, the all sums of any $\left\lfloor \frac{d-1}{2} \right\rfloor$ columns of **H** are different. Hence for a double error correcting code ($d_{\min} \geq 5$) any two columns of **H** will produce a different syndrome ($h_i+h_j \neq h_k+h_l$, $i \neq j, k \neq l$). Thus, the code $C=\{h_0,h_1,...,h_{n-1}\}$ is 2-Robust since for any error e, $|\{h_i \mid h_i \in C, h_i \oplus e \in C\}| \leq 2$.

Corollary 1

A 2-Robust error-detecting code with a codeword of all zeros will form the columns of a check matrix of a double error-correcting binary linear code (omitting the zero codeword). Any 2-Robust error-detecting code can be modified such that the code will form the columns of the check matrix of a double error-correcting linear code.

Proof:

Any 2-Robust error detecting code already has the restriction that no more than one pair of codewords will have the same sum. If a codeword of all zeros exists then no codeword will equal the sum of two codewords (except the sum of the zero and codeword and any other codeword) since that would mean that there were in fact two pairs with the same sum which would be a contradiction of the definition of a 2-Robust error-detecting code. Likewise, any 2-Robust code can be shifted such that it will contain the zero codeword.□

Example 2

The check matrix of a double error correcting binary BCH code has the following check matrix:

$$H=\begin{bmatrix} 1 & \alpha & \alpha^2 & ... & \alpha^{n-1} \\ 1 & \alpha^3 & (\alpha^2)^3 & ... & (\alpha^{n-1})^3 \end{bmatrix}$$ where *n=k+r, $n=2^r-1$ and α is a primitive element of $GF(2^r)$*

Thus, the codewords of the corresponding nonlinear Robust code generated by the check-matrix have the form (x,x^3), $x \in GF(2^k)$. This construction based on the cubic has been shown 2-Robust in [10].

In addition to being invariant to the addition and multiplication, Robust codes based on perfect nonlinear transformations are subject to additional linear transformations which will preserve some Robust properties and can be useful adding Robustness while minimizing encoding hardware for computation channels.

Theorem 3

Let $f: GF(2^k) \rightarrow GF(2^r)$ have perfect nonlinearity and let $l: GF(2^r) \rightarrow GF(2^r)$ be linear onto function. The set of words in the form of $(x, l(f(x)))$ form a uniformly Robust code.

Proof

The composition $l \circ f$ is a function from $GF(2^k)$ to $GF(2^r)$ with perfect nonlinearity and so by Theorem 1 the resulting set of words form a uniformly Robust code. □

The opposite composition of perfect nonlinear function and linear function does not preserve perfect nonlinearity and hence does not guarantee Robustness but the transformation has some useful properties useful for practical considerations as outlined in the following theorem.

Theorem 4

Let $f : GF(2^r) \to GF(2^r)$ have perfect nonlinearity and let $l : GF(2^k) \to GF(2^r),\ r \le k$ be a linear onto function. The set of words in the form $(x, f(l(x)))$ form a code with 2^{k-r} undetectable errors.

Proof

By Theorem 3 the set of words in the form $(l(x), f(l(x)))$ forms a uniformly Robust code of length 2r. For this code the only undetectable error is the zero error. Since l is a linear onto function it maps 2^{k-r} words from $GF(2^k)$ to the r-bit zero vector. Hence the code with words in the form $(x, f(l(x)))$ has a total of 2^{k-r} undetectable errors.□

Based on the previous theorem, the number of undetectable errors can be greatly reduced by applying a perfect nonlinear transformation to the redundant portion of the systematic linear code. It is important to note that the additional transformation does not create any additional undetectable errors. The undetectable errors in the Robustly modified linear code form a *k-r* dimensional subspace of the corresponding linear code.

Some specific constructions and properties of Robust codes based on perfect nonlinear and almost perfect nonlinear power functions were previously shown in [10]. The codes from [10] had the following constructions

$C = \{(y, R(y)) \mid y \in GF(2^k), Py \in GF(2^r), R(y) = (Py)^3\}$

and

$C = \{(y, R(y)) \mid y \in GF(q^k), Py \in GF(q^r), R(y) = (Py)^2, q > 2\}$. *where P is a (kxr) matrix*

For the case when is *P* is an identity matrix the resulting $(n = 2k, k)$ codes were proven to be Robust.

III. BENEFITS, AND APPLICATIONS OF ROBUST CODES

Uniformly Robust codes provide for *equal error detection against all errors* (assuming a uniform distribution of data messages) *independently of error distributions*. There are generally no assumptions about the techniques (in terms of error probabilities) that an attacker will use. In some situations due to nonlinear (and generally quadratic complexity) encoding and decoding procedures uniformly Robust codes might not be completely practical in consideration of the overhead. But the use of Robust constructions can still have benefits. Specifically, as shown in Theorem 4, application of nonlinearity to linear codes decreases the number of undetectable errors and redistribute the error detecting power of the original linear code reducing the potential weaknesses.

For example, consider the *(2k,k)* duplication code which has 2^k undetectable errors. These undetectable errors are predictable and independent of the data. In such a system an undetectable error can be easily injected by injecting the same fault into the two copies of the device. Furthermore, although the average probability of missing an error is 2^{-k} (assuming all errors are equiprobable) the actual probability of detection depends heavily on the class and type of errors injected. This limitation can be easily visible in the error detection profile for the k= 8 shown in Figure 2A. Considering even the simple different classes of errors based on multiplicity (which can generally be easily controlled by an attacker) it can be seen that the detection is not uniform and an attacker can greatly increase his chances of injecting an undetectable error by considering errors of multiplicity two. In contrast, the Robustly modified inversion based duplication code with the same *k* defined as $C = \{(x, x^{-1}) \mid x \in GF(2^k)\}$ where (x^{-1} is the multiplicative inverse in $GF(2^k)$) has no undetectable errors. We note that such a code can easily be constructed from the regular duplication code by taking a perfect nonlinear transformation of the redundant bits in the respective field. As can be seen in Figure 2B, the Robust code results in a redistribution of the error detecting power and has almost equal protection for any error classes providing a provable and reliable level of security against an unpredictable fault attack. The Robust codes offer a significant improvement over linear codes with respect to the three criteria listed in the introduction by decreasing the number of undetectable errors without changing the redundancy of the code.

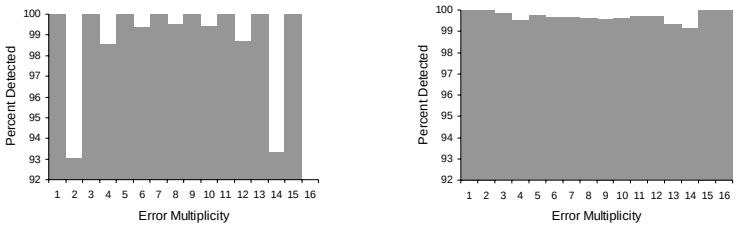

Figure 2. Error detection profile of (A) a linear k=r=8 duplication code and a (B) Robust nonlinear cubic k=r=8 duplication code $C=\{(x,x^3) \mid x \in GF(2^8)\}$

An additional direct consequence of the nonlinearity of the Robust codes is that the error detection for these codes is data dependent. That is, unlike in a linear code where each error is detected or missed independently of the data, the errors which are missed for Robust codes are generally dependent on the data and the set of missed errors is different for each data input. This property has the advantage in that it makes the set of necessary errors harder to determine and inject (provided that an attacker has such a capability) since the data (output of the cryptographic device) depends on the secret key. This property combined with additional masking, blinding, or randomization methods can add to the difficulty of determining errors which would be missed by making them non deterministic and hard to determine even if an attacker has an ability to encrypt and decrypt chosen plaintexts and ciphertexts.

Finally the proposed error detecting codes have the advantage of an increased probability of detecting jamming attacks and permanent failures which result in repeating errors. For linear codes, if a hardware failure (say, from tampering) produces a fault within a circuit which results in a repeated error which is a codeword the fault will always be undetected. For the Robust code, which has data dependent detection, any repeating error, will eventually be detected.

IV. ROBUST HARDWARE ARCHITECTURES

The general architecture used for protecting hardware devices based on the use and analysis of error-detecting codes is shown in Figure 3. The architecture is based on adding redundancy around an original device to create data redundancy which can be used to verify data integrity and the correct operation of the device. The architecture is composed of three major hardware components: original hardware, redundant hardware for predicting the r-bit signature of the original device and an error-detecting network (EDN) which verifies the predetermined relationship of the output of the original device and the signature of the Predictor.

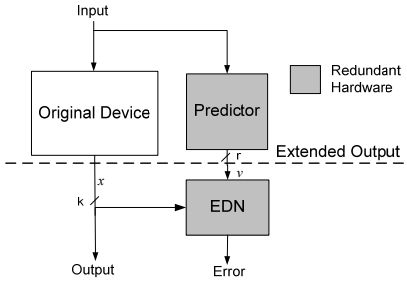

Figure 3. General architecture for protection of hardware with error-detecting codes

By defining an appropriate error-detecting code and implementing the suitable redundant function in the Predictor a desired level of error detection capability can be guaranteed for the desired output of the Original Device.

However, the analysis and design of the protection based on the error-detecting code of the Extended Output is based on the assumption that the hardware of the EDN is fault/error free. This requirement usually dictates that in practical implementations the hardware within the EDN is small compared to the protected device since the EDN needs extra protection and is usually implemented as a totally-self-checking design or other high cost protection methods in order to ensure the fault free operation [17].

The protection approach can be characterized by a pair of trivial implementations which represent the two extremes of the architecture and which also demonstrate the large space and degree of freedom for optimization and

redesign which can be applied to specific designs. The first trivial approach is based on making the design and function of the Predictor identical to that of the Original Device. This approach has the advantage in that it allows for a minimal EDN. The disadvantage and impracticality of this extreme arises due to the overall overhead, which is over 100%, and the fact that corresponding error detecting code created at the Extended Output is a *linear* duplication error-detecting code. As shown in the previous section, a linear replication code has many weak properties which might make it not suitable for protection against an active attacker as it can be predictably bypassed.

The second trivial approach which represents the other extreme of the possible form of the architecture is based on the design of the Predictor with an identity function and an EDN which performs an inverse function of the Original Device (when possible) [11]. There is a natural inverse relationship between encryption and decryption in symmetric ciphers such as AES and the approach to the architecture can and has been easily implemented. If the function of the block cipher is assumed to be a perfectly or a near perfectly nonlinear function then the Extended Output of the device, which consists of the input and output of the cipher can be viewed as Robust based on Theorem 1. The Robust aspect of the approach is exactly what is desired in an implementation. However, for this analysis of Robustness to be valid the EDN hardware is assumed to be fault free. This assumption cannot be guaranteed due to the large hardware overhead of the EDN. As a result although the analysis of the error detecting code appears to indicate Robustness the property cannot be guaranteed.

Between the two extremes there is room for optimization and reduction of overhead while providing for a level of Robustness. We offer a design approach which aims to provide compromise between the two ends of the somewhat impractical linear and completely Robust ends of the spectrum. The design incorporated into the protection of AES is based on the combination of a linear prediction function followed by a nonlinear transformation which adds Robustness to the respective code and reduces the number of undetectable errors (Theorem 4). The reduction of the size of the signature reduces the size of the EDN allowing analysis of the level of protection based on the properties of the error detecting code at the Extended Output.

V. APPLICATION OF ROBUST CODES TO SECURE IMPLEMENTATIONS OR AES

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 proceeded by a round key addition. In each of the nine typical rounds there are four transformations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The ShiftRows and AddRoundKey transformations are trivial in hardware implementations as they require only a permutation or a small number of XOR gates. The MixColumns transformation is the second most complex. It is linear over *GF(2)* and can be implemented with a series of XOR operations. The SubBytes transformation is the most costly and involves two operations: inversion in $GF(2^8)$ followed by an affine transform which involves a matrix multiplication *M* over $GF(2)$, followed by addition of a constant vector τ. With the exception of inversion, all other transformations and operations are linear (Figure 3). That is, they can all be implemented using XOR gates only. We omit the details of the individual transformations which can be found in [2].

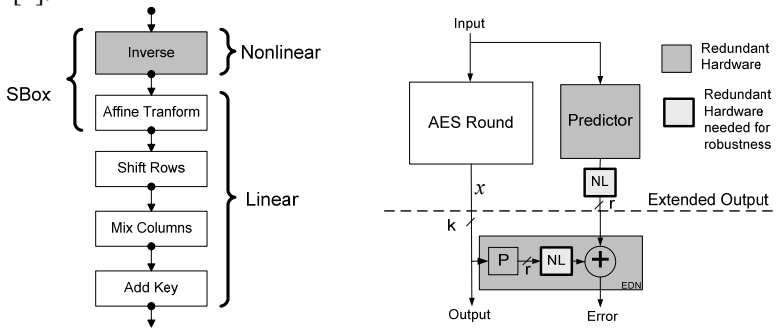

Figure 3. (A) Transformation involved in one typical round of AES (B) General Robust Architecture with nonlinear transforamtions

The combination of linear and nonlinear transformations found in AES makes it difficult to efficiently generate and propagate prediction or parity check bits.

For the AES it is possible to implement an efficient linear prediction scheme which can be used to generate a r=32-bit signature which is suitable for hardware implementations [16]. For four bytes of the output, the predictor predicts one byte, $L(j)$ $(j = 0,1,2,3)$ for a total of redundant bytes of the 128-bit original output. For encryption this method simplifies the prediction function by eliminating the MixColumns transformation.

The output of the linear predictor, $L(j)$, is a 4-byte word which is linearly related to the output of one round of AES. The function of $L(j)$ with respect to *Out(i,j)* can be written as:

$L(j) = \bigoplus_{i=0}^{3} Out(i, j)$ where $j \in \{0,1,2,3\}$.

Thus, the following expressions are valid for AES:

$$\begin{aligned} L(0) = {} & 01 \bullet Sub(In(0,0)) \oplus 03 \bullet Sub(In(1,0)) \oplus Sub(In(2,0)) \oplus Sub(In(3,0)) \oplus \\ & Sub(In(0,0)) \oplus 02 \bullet Sub(In(1,0)) \oplus 03 \bullet Sub(In(2,0)) \oplus Sub(In(3,0)) \oplus \\ & Sub(In(0,0)) \oplus Sub(In(1,0)) \oplus 02 \bullet Sub(In(2,0)) \oplus 03 \bullet Sub(In(3,0)) \oplus \\ & 03 \bullet Sub(In(0,0)) \oplus Sub(In(1,0)) \oplus Sub(In(2,0)) \oplus 02 \bullet Sub(In(3,0)) \\ & \oplus RK(0,0) \oplus RK(1,0) \oplus RK(2,0) \oplus RK(3,0) \end{aligned}$$

which simplifies to

$$\begin{aligned} L(0) = {} & Sub(In(0,0)) \oplus Sub(In(1,0)) \oplus Sub(In(2,0)) \oplus Sub(In(3,0)) \\ & \oplus RK(0,0) \oplus RK(1,0) \oplus RK(2,0) \oplus RK(3,0), \end{aligned}$$

where $\bullet$ is multiplication in $GF(2^8)$, $In(i, j)$ is a text input byte to the round, $RK(i, j)$ is one byte round key, and $Sub(In(i, j))$ is the SubBytes transformation on the byte $In(i, j)$ as defined in the AES standard [2].

Since $Sub(In(i, j)) = M(In(i, j)^{-1}) \oplus \tau$, We have $L(0)$:

$$L(0) = M(In(0,0)^{-1} \oplus In(1,1)^{-1} \oplus In(2,2)^{-1} \oplus In(3,3)^{-1}) \oplus RK(0,0) \oplus RK(1,0) \oplus RK(2,0) \oplus RK(3,0).$$

Extending the procedure to the rest of the bytes of the predictor for encryption yields:

$$L(1) = M(In(0,1)^{-1} \oplus In(1,2)^{-1} \oplus In(2,3)^{-1} \oplus In(3,0)^{-1}) \oplus RK(0,1) \oplus RK(1,1) \oplus RK(2,1) \oplus RK(3,1),$$

$$L(2) = M(In(0,2)^{-1} \oplus In(1,3)^{-1} \oplus In(2,0)^{-1} \oplus In(3,1)^{-1}) \oplus RK(0,2) \oplus RK(1,2) \oplus RK(2,2) \oplus RK(3,2),$$

$$L(3) = M(In(0,3)^{-1} \oplus In(1,0)^{-1} \oplus In(2,1)^{-1} \oplus In(3,2)^{-1}) \oplus RK(0,3) \oplus RK(1,3) \oplus RK(2,3) \oplus RK(3,3).$$

Thus the Predictor which outputs a predicted 32-bit output which is linearly related to the original output of an AES round of encryption is much simpler and involves only multiplicative inverse operations and XOR additions. The MixColumns transformation is not present in the Predictor transformations and the number of matrix multiplications of the Sbox transformations are greatly reduced. The predictor can be combined for prediction of decryption as well as key-expansion operations. In FPGA implementations the Predictor can be implemented such that it is only about 50% the size of a typical round of AES while providing a 32-bit redundant output.

In the non-Robust implementation the Error Detecting Network (EDN) compresses the 128 bits into r=32 by bitwise XOR to match the output of the Predictor followed by a comparison. Without adding Robustness the linear (160, 128) linear code has $2^k = 2^{128}$ undetectable errors and the probability of missing an error (assuming all errors are equiprobable) is $2^{-r} = 2^{-32}$. This protection offered by the linear protection is not optimal with respect to the criteria defined in the introduction. By applying a nonlinear transformation the number and hence the probability of injecting and undetectable error can be greatly reduced.

The linear protection can be extended and made more Robust by the addition of two nonlinear transformations, one after the Predictor and one in the EDN as shown in Figure 3B (represented by the NL block). The nonlinear blocks can be, for example a cubic in the respective field. By cubing (in $GF(2^r)$) the output of the Predictor the respective code created at the Extended Output of the device (Figure 3B) reduces the number of undetectable errors

to 2^{k-r} and the probability of injecting an undetectable error is reduced to 2^{-2r} without increasing the redundancy of the code.

An iterative Robust architecture which repeats one round of AES was implemented on a Xilinx XCV1000 FPGA using a cube in $GF(2^r)$ as the nonlinear transformation. For the final design a signature with r=28 bits was chosen since the respective field has a good trinomial the cubic operations can be implemented efficiently. The final design where r=28 resulted in a 77% hardware overhead and a probability of injection of an undetectable error into the architecture is 2^{-56}.

The (156,128) Robust and linear code was simulated for random errors (assuming all errors and messages to be uniformly distributed) by randomly injecting errors into the extended output of the device. The resulting Robust code showed improved detection over the linear code. After injecting almost 30 billion errors no undetectable errors were encountered while for the linear case 118 undetectable errors were found (Table 1) which is matches the expected values of the linear and Robust codes.

Table 1. Results of simulations of symmetrical errors.

	ERROR PATTERNS INJECTED	ERRORS MISSED AFTER 5 RANDOM INPUTS
LINEAR PROTECTION	29.7 billion	118
ROBUST CUBIC PROTECTION	29.7 billion	0

The method for modifying an architecture based on linear codes into a robust architecture codes requires an overhead for computation of a perfect or almost perfect nonlinear transform in $GF(2^r)$, which is generally of the order $O(r^2)$. Since large *r* may be necessary to provide for a sufficiently high error-detecting probability the use of one large device which takes the nonlinear transformation of all of the *r*-redundant bits might not be practical.

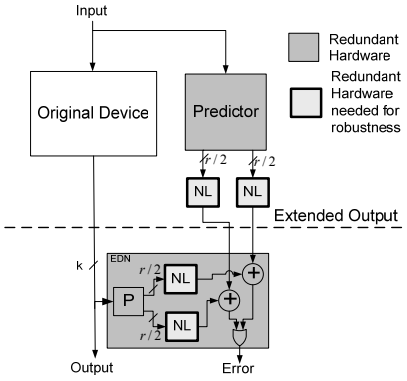

Figure 4. Optimized architecture, the nonlinear transformation is split into *t=2* separate modules

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 nonlinear transformation for all *r*-bit vectors, it is possible to divide the one large nonlinear transformation into disjoint smaller nonlinear transformations while retaining many of the robust properties outlined earlier. That is, we can replace nonlinear transformations in $GF(2^r)$ by t s-bit disjoint nonlinear transformation in $GF(2^{\frac{r}{t}})$ to produce the nonlinear *r* bit output ($r = ts$). Thus, instead of having two *r*-bit nonlinear transformations in $GF(2^r)$ for the whole design, there could be $2t$ nonlinear transformations in $GF(2^{\frac{r}{t}})$ as it is presented in Figure 4 for *t=2*. Since the number of two input gates to implement the nonlinear transformation is proportional to the square of the number of bits at its input, a modification where *t=2* results in roughly 50% decrease of an overhead associated with the architecture based on robust codes. This also results in a slight decrease in the level of robustness and an in introduction of errors which are detected with different probabilities.

The division of the nonlinear signature results in the creation of additional classes of errors which detected with different probabilities depending on the number of different signatures they affect. Table 2 shows the redistribution of errors among the additional classes for t s-bit signatures if an multiplicative inverse is taken as the nonlinear transformation and r is odd.

Table 2. Redistribution of errors as function of the number of blocks t of the signature when r is odd

#of blocks	Number of errors missed with probability p				
	p=1 (undetec table)	p=0 (always detected)	$p=2^{-s+1}$	$p=2^{-2(s+1)}$	$P=2^{-i(s+1)}$
Linear *(t=r)*	2^k	$2^{k+r}-2^k$	0	0	0
$t=1$ (robust)	N_1	N_2	N_3	0	0
$t<\frac{r}{2}$ (robust)	$(N_1)^t$	$\sum_{i=1}^{t}\binom{t}{i}N_2^{\,i}(2^s-N_2)^{t-i}$ $=(2^s)^t-(2^s-N_2)^t$	$tN_3(N_1)^{t-1}$	$\binom{t}{2}(N_3)^2(N_1)^{t-2}$	$\binom{t}{i}(N_3)^i(N_1)^{t-i}$
$t\geq\frac{r}{2}$	2^k	$2^{k+r}-2^k$	0	0	0

where $N_1=2^{\frac{k}{t}-s}$, $N_2=2^{\frac{k+r}{t}-1}+2^{\frac{k}{t}-1}-2^{\frac{k}{t}-s}$, $N_3=2^{\frac{k+r}{t}-1}-2^{\frac{k}{t}-1}$

The splitting of the signature has several effects. Depending on the number of blocks, *t,* there is a redistribution of errors and a difference in the level of robustness. The maximum robustness is achieved with no divisions when *t=1*. With an increasing number of blocks the robustness of the resulting code is reduced. As the number of blocks, *t,* increases, the number of undetectable errors increases exponentially. Likewise, the number of classes of errors increases linearly as *t* increases. Figure 6 demonstrates the increase of robustness, or uniformity or error coverage, as the number of blocks in a *r*-bit signature decreases for duplication where *k=r=8*. The level of robustness, or uniformity of error detection, increases as the number of signature divisions *t* decreases providing a tradeoff between overhead and robustness.

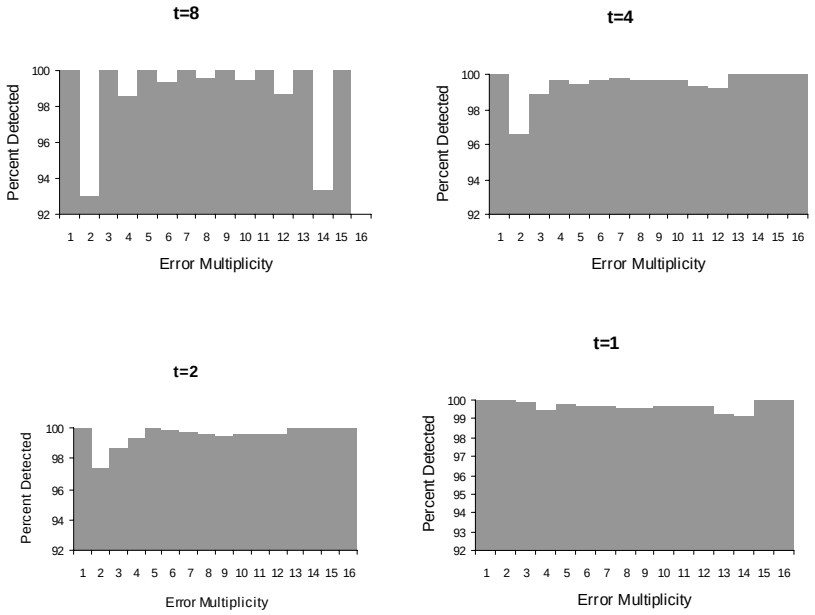

Figure 5. The effect of splitting the *r*-bit signature into *t* disjoint inverses on robustness where *k=r=8*.

VI. CONCLUSIONS

The protection methods which have been adapted from tradition fault-tolerant architectures are not optimal for the protection of cryptographic hardware susceptible to fault analysis attacks. The protection in the traditional architectures which is usually based on linear error detecting codes is not uniform and the level of protection they provide depends largely on the type of error that is considered. We presented a method and formal description of protection based on nonlinear systematic robust codes which can provide for uniform protection against all errors without making any assumptions about the error and capabilities of an attacker. The Robust constructions are based on perfect or almost perfect nonlinear functions which are an integral part of many cryptographic algorithms. The Robust constructions can be applied to existing architectures based on linear error-detecting codes to redistribute their error detecting power and reduce the number of undetectable errors.

We presented an example application of the Robust construction to an implementation of the hardware for Advanced Encryption Standard, secure against DFA attacks.

REFERENCES

[1] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems", CRYPTO 97, LNCS 1294, pp.513-525

[2] FIPS PUB 197: "Advanced Encryption Standard", http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[3] C.N. Chen and S.M. Yen. “Differential Fault Analysis on AES Key Schedule and Some Countermeasures”. ACISP 2003, LNCS 2727, pp18-129, 2003.

[4] P. Dusart, G. Letourneux, O. Vivolo, “Differential Fault Analysis on AES”. Cryptology ePrint Archive, Report 2003/010. Available: http://eprint.iacr.org/2003/010.pdf

[5] C. Giraud. “DFA on AES”. Cryptology ePrint Archive, Report 2003/008. Available: http://eprint.iacr.org.

[6] J. Blömer and J.P. Seifert. "Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)". Financial Cryptography 2003: pp. 162-181.

[7] J.J. Quisquater and G. Piret. “A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD”. CHES 2003, LNCS 2779, pp 77-88, 2003.

[8] M. G. Karpovsky, P. Nagvajara, "Optimal Robust Compression of Test Responses," IEEE Trans. on Computers, Vol. 39, No. 1, pp. 138-141, January 1990.

[9] M. G. Karpovsky, P. Nagvajara, "Optimal Codes for the Minimax Criterion on Error Detection," IEEE Trans. on Information Theory, November 1989.

[10] M.G.Karpovsky and A. Taubin, "A New Class of Nonlinear Systematic Error Detecting Codes", *IEEE Trans Info Theory*, Vol 50, No.8, 2004, pp.1818-1820

[11] R. Karri, K. Wu, P. Mishra, and Y. Kim. "Concurrent Error Detection of Fault Based Side-Channel Cryptanalysis of 128-Bit Symmetric Block Ciphers". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.21, No.12, pp. 1509-1517, 2002

[12] R. Karri, G. Kuznetsov, M. Gössel. "Parity-Based Concurrent Error Detection of Substitution-Permutation Network Block Ciphers". *In Proc. of CHES 2003*. pp.113-124.

[13] 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, 2003

[14] C. Carlet and C. Ding, "Highly nonlinear mappings," *Journal of Complexity*, vol. 20, pp. 205-244, 2004

[15] S. P. Skorobogatov. “Semi-Invasive Attacks – a New Approach to Hardware Security Analysis”. Technical Report, University of Cambridge. Number 630.

[16] M. G. Karpovsky, K. J. Kulikowski, and A. Taubin, "Robust Protection Against Fault-Injection Attacks of Smart Cards Implementing the Advanced Encryption Standard" Dependable Systems and Networks (DSN'04), July, 2004

[17] P. K. Lala. “Self-Checking and Fault-Tolerant Digital Design”. Morgan Kaufmann Publishers, 2001.