Robust Protection against Fault-Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard*

Mark Karpovsky, *Fellow, IEEE*, Konrad J. Kulikowski, Alexander Taubin, *Senior Member, IEEE*
Reliable Computing Laboratory
Department of Electrical and Computer Engineering
Boston University
8 Saint Mary's Street, Boston, MA 02215
{markkar, konkul, taubin}@bu.edu

Abstract

We present a method of protecting a hardware implementation of the Advanced Encryption Standard (AES) against a side-channel attack known as Differential Fault Analysis attack. The method uses systematic nonlinear (cubic) robust error detecting codes. Error-detecting capabilities of these codes depend not just on error patterns (as in the case of linear codes) but also on data at the output of the device which is protected by the code and this data is unknown to the attacker since it depends on the secret key. In addition to this, the proposed nonlinear (n,k)-codes reduce the fraction of undetectable errors from 2^{-r} to 2^{-2r} as compared to the corresponding (n,k) linear code (where n-k=r and k>=r). We also present results on a FPGA implementation of the proposed protection scheme for AES as well as simulation results on efficiency of the robust codes.

1. Introduction

Today's information security engineer is faced with the problem of building a trustworthy system from untrustworthy components. Security experts claim that the only workable solutions to date demand some minimal number of trustworthy components. These trustworthy components are relied on for ensuring overall system security by providing services such as authentication, encryption/decryption, cryptographic tokens and so on [1].

Security is typically provided at the level of software (cryptographic algorithms). Traditional cryptographic protocol designs assume that input and output messages are available to attackers, but other information about the keys is not available. However, during the last seven years a new class of attacks against cryptographic devices has become public [2]. These attacks exploit easily accessible information like power consumption, running time, input-output behavior under malfunctions, and can be mounted by anyone using low-cost equipment. These *side-channel attacks* amplify and evaluate leaked information with the help of statistical methods and are often much more powerful than classical cryptanalysis. Examples show that a very small amount of side-channel information is enough to completely break a cryptosystem [3]. While many previously-known cryptanalytic attacks can be analyzed by studying algorithms, side-channel attacks vulnerabilities result from electrical behavior of transistors and circuits of an implementation. This ultimately compromises cryptography and shifts the top priority in cryptography from the further improvement of algorithms to the prevention of such attacks by reducing variations in timing, power and radiation from the hardware [4], reduction of observability of system behavior after fault injection [5], and theoretical extension of the current mathematical models of cryptography to the physical setting which takes into consideration side-channel attacks [6].

In this paper we focus on the side-channel attacks known as Differential Fault Analysis (DFA) [2] attacks. DFA was first proposed in 1997 by E. Biham and A. Shamir [7] as an attack on DES. The attacks have since been applied to AES by others [8,9,10,11]. DFA attacks are based on deriving information about the secret key by examining the differences between a cipher resulting from correct operation and a cipher of the same initial message resulting from faulty operation.

Several research groups suggest concurrent error detection procedures as a hardware countermeasure against fault injection based cryptanalysis. Karri et al. [12]

* This work was partially supported by the Community Technology Fund of Boston University

propose to add circuitry to perform decryption, in parallel with the encryption (with various possible levels of granularity) and compare them with the input value to ensure that no error has occurred. These solutions have different detection time latencies and hardware costs and, in general, exhibit a large cost close to that of duplication either in space or in time. It is clear that not all-possible attacks have been taken into account. The conclusion of [12] states that it is assumed that both encryption and decryption modules are not simultaneously under attack or faulty, that is not very realistic for example for smart card applications.

The fault-detecting scheme for AES from [13] is based on one-dimensional parity codes. They propose associating one redundant parity bit with each byte of the state matrix. It provides for detection of errors involving an odd number of bits in a byte. Unfortunately, the attacker can still be successful if only even number of errors in a byte element of state matrix is injected by the attacker .

In our design, after a DFA attack is detected the device implementing AES disables itself. We assume that the number of natural faults which can occur in a life span of a device is much less than the number of faulty ciphertexts needed for a realistic DFA attack. The disabling circuitry can be composed of a simple counter which counts the number of errors detected. When a predetermined threshold is reached the device will clear the secret key from its memory thus preventing any further attacks. This count threshold can be adjusted depending on the operating environment and expected life span of the device. This method is only as effective as the error detecting codes used. One of the most important criteria for this method is that the error coverage of the code is as large as possible while maintaining a reasonable hardware overhead.

We thus present a new class of systematic **nonlinear robust** codes in Section 3 and propose a robust protection scheme against such attacks. We will use systematic nonlinear robust codes for detection of DFA attacks. The proposed nonlinear robust codes can be used to extend the error coverage of linear codes without increasing their redundancy. The hardware overhead of this method is less than the overhead that would be necessary if the error coverage of the linear code was increased by increasing the code's redundancy. We will use stuck-at fault and bit flip error models to justify the use of the nonlinear robust codes.

We note that optimal *nonsystematic* robust codes have been proposed in [14, 15] but these codes require rather complicate encoding and decoding procedures, which prohibits application of these codes for detection of DFA attacks.

For the proposed robust codes the probability of error detection depends not only on the error pattern (as in the case of linear codes) but also on the data itself. If all the data vectors and error patterns are equiprobable, then the probability of injecting an undetectable error if the device is protected by our robust codes is 2^{-2r} versus 2^{-r} if the device is protected by any linear code with the same r (r is a number of redundant bits which are added for data protection).

For brevity, we omit the discussion of AES. The reader may consult [16] for detailed specification. In Section 2 we give a detailed description of the fault model used, in Section 3 we present the *systematic* nonlinear robust codes with simple encoding and decoding procedures. Section 4 has a description of a general architecture, which can be used to protect AES with the presented robust codes. Section 5 has a detailed description of a FPGA implementation of a robust protected AES core. In Section 6 we summarize the size and overhead statistics of our FPGA implementation. Section 7 shows the results of our simulations on probabilities of detecting a DFA attack, which support our initial calculations. Finally, in Section 8 we present advantages of using the systematic nonlinear robust codes presented for the protection of AES versus other countermeasures.

2. Fault model

We refer to a fault as a physical malfunction of a part of a circuit, for example a wire being stuck-at zero, or an output of a gate being stuck-at one. A fault is what is directly created by an attacker. Faults can generally be induced into a device by subjecting it to abnormal conditions. Voltage spikes, clock glitches, extreme temperatures, radiation, eddy currents, and light can all cause faults. However, with all of these, with the exception of light [17, 18], there is no control as to the location, and type of a fault which will be induced, these are sometimes called probabilistic attacks. An error is a manifestation of fault at the output of the device. An error is the difference (componentwise XOR) between the correct and distorted outputs of the device.

In this paper, we consider protection against a probabilistic attack. In this type of an attack, the attacker has little or no control as to the location or type of fault that is injected. This attack does not necessitate chip depackaging or specialized equipment, and as a result it is one of the most accessible attacks. Regardless of where a fault occurs, the fault is only meaningful to an attacker if it manifests itself as an error at the output of the device.

Thus detecting a fault attack is equivalent to detecting the corresponding error in the output of the device, or in the case of AES between each round. In addition, because a probabilistic attack has little control over the location and timing of the faults, and hence the errors which occur, we further assume that the errors resulting from an attack are uniformly distributed and remain constant for several different text inputs.

3. Systematic nonlinear robust codes

Let V be a binary linear *(n,k)*-code with $2k \geq n$ and check matrix $H = [P|I]$ where I is an (rxr) identity matrix and P is an *((n-k) x k)* matrix of rank $n-k=r$ over *GF(2)* [19]. Then for any message, error e is not detected iff $e \in V$. As it will be shown below this linear *(n,k)*-code V can be modified into a nonlinear robust systematic *(n,k)*-code C_V such that set E of undetected errors for C_V is a *(k-r)*-dimensional subspace of V ($|E| = 2^{k-r}$ instead of 2^k for V).

Theorem I

Let $C_V = \{(x,w) | x \in GF(2^k), w = [Px]^3 \in GF(2^r)\}$.

Then the set $E = \{e | y \oplus e \in C_V \text{ for all } y \in C_V\}$ of non-detected errors for C_V is a *(k-r)*-dimensional subspace of V, and from the remaining $2^n - 2^{k-r}$ errors $2^{n-1} + 2^{k-1} - 2^{k-r}$ are detected with probability 1 for any message and $2^{n-1} - 2^{k-1}$ are detected with probability $1 - 2^{-r+1}$. (All the messages assumed to be equiprobable).

Proof:

Error (e_x, e_w) *($e_x \in GF(2^k)$, $e_w \in GF(2^r)$)* is not detected for message $(x, [Px]^3)$ from C_V iff:

$$[P(x \oplus e_x)]^3 = [Px]^3 \oplus e_w \quad \textbf{(1)}$$

(All computations in **(1)** are in $GF(2^r)$)

or

$$[Px]^2 [Pe_x] \oplus [Px][Pe_x]^2 \oplus [Pe_x]^3 \oplus e_w = 0 \quad \textbf{(2)}$$

It follows from (2) that $e = (e_x, e_w)$ is not detected for any x iff $Pe_x = e_w = 0$, and $E = \{(e_x, e_w) | Pe_x = e_w = 0\}$ is a *(k-r)*-dimensional subspace in $V = \{(x,w) | w = Px\}$.

If $Pe_x = 0$ and $e_w \neq 0$, then (e_x, e_w) is detected by C_V for any x. There are

$$N_1 = 2^k - 2^{k-r} \quad \textbf{(3)}$$

errors satisfying this condition.

For any given (e_x, e_w) such that $Pe_x \neq 0$ quadratic equation (2) has 2 solutions for Px iff

$$Tr([Pe_x]^{-3} ([Px]^3 \oplus e_w)) = Tr(1) \oplus Tr([Pe_x]^{-3} e_w) = 0 \quad \textbf{(4)}$$

and has 0 solutions iff

$$Tr(1) \oplus Tr([Pe_x]^{-3} e_w) = 1 \quad \textbf{(5)}$$

where $Tr(y)$ is the trace of y in $GF(2^r)$ [19].

Since out of $2^n - 2^k$ errors $e = (e_x, e_w)$ such that $Pe_x \neq 0$

$$N_2 = 2^{n-1} - 2^{k-1} \quad \textbf{(6)}$$

satisfy (4), we have from (3) and (5) for a number, N, of errors which are detected for any x $N = N_1 + N_2 = 2^{n-1} + 2^{k-1} - 2^{k-r}$.

Finally, the remaining $2^{n-1} - 2^{k-1}$ errors satisfying (4) are detected with probability $1 - 2^{-r+1}$. □

Table 1. Comparison of the proposed robust codes and corresponding linear codes

	ROBUST NONLINEAR	LINEAR NOT ROBUST
Number of undetectable errors.	2^{k-r}	2^k
Number of errors detected with probability of 1	$2^{n-1} + 2^{k-1} - 2^{k-r}$	$2^n - 2^k$
Number of errors detected with probability $1 - 2^{-r+1}$	$2^{n-1} - 2^{k-1}$	0

The transition from linear code V to the corresponding non-linear code C_V requires only addition of two cubic networks. Each cubic network increases the complexity of encoding and decoding by $O(r^2)$. Thus replacing linear *(n,k)* code ($k \geq n/2$) by a cubic robust code with the same parameters results in a reduction of the size of the space of undetected errors from 2^k to $2^{k-r} = 2^{n-2r}$. The properties of the proposed robust codes versus linear are summarized in Table 1.

In the above theorem and proof we had chosen a cubic network as the nonlinear function. We note that a square in the respective field would not work. Other functions are also possible. One alternative is to use a multiplicative inverse in $GF(2^r)$ or taking a higher power. The alternative nonlinear functions result in a larger hardware overhead or reduced error coverage. Thus, the cubic function was preferred since it is in general of a lower complexity and a results in higher error coverage.

4. General architecture

Robust codes can be used to extend the error coverage of any linear prediction scheme for AES. Only two extra cubic networks are needed, one in the extended device, and one in the Error Detection Network (EDN). The architecture of AES with robust protection is presented in Figure 1.

In the architecture in Figure 1 a single linear predictor is assumed for the encryptor, decryptor, and key expansion. The same architecture can be extended to architectures, which would have separate linear predictors for all the devices. (Note that in this context a linear predictor is such that it generates a signature, which is a linear combination of the outputs of the round. It does not

mean that the predictor contains only linear elements. It could in fact contain nonlinear elements just as long as its output is linear with respect to the output of the round.) It is the *r*-bit signature of the linear predictor, which is cubed in $GF(2^r)$ to produce an *r*-bit output, which is nonlinear with respect to the output of the round.

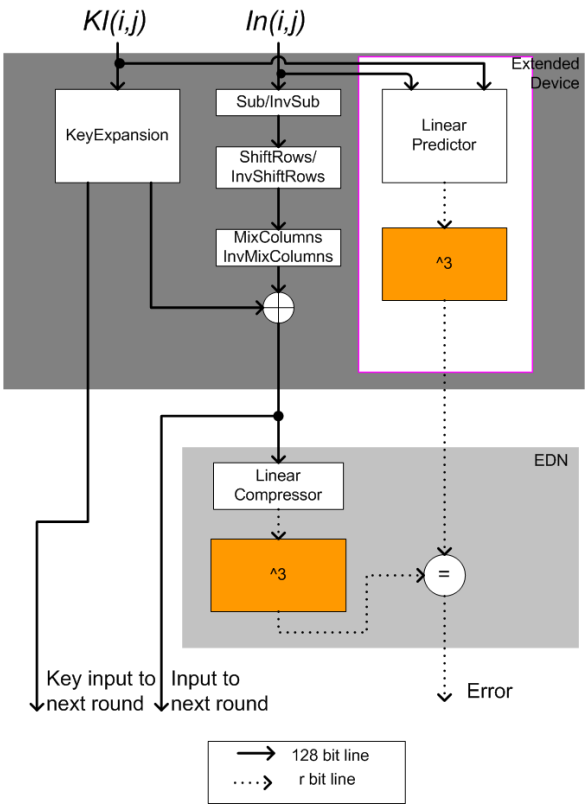

Figure 1. Robust architecture for one round of AES.

For the robust architecture we have designed a linear predictor which can be used to generate a r_L=32-bit signature. The proposed linear predictor offers a relatively compact design, which allows for easy hardware sharing for encryption and decryption prediction. The single predictor is designed so that it protects the encryptor/decryptor as well as key-expansion. The complete design of the linear predictor can be found in the next section.

5. Detailed design of the linear predictor

The output of the linear predictor is linearly related to the output of the round of AES (see Figure 2). Specifically, each byte of the linear predictor's output $L'(j)$ is equivalent to the componentwise XOR of four bytes of the output of a round (see Figure 3 for summary of the notations used):

In this method the function of each byte of $L'(j)$ no longer contains the MixColumns transformation [16]. As a result, the linear predictor is greatly simplified since it no longer needs to perform multiplications associated with the MixColumns/InvMixColumns (see [16]). The details of the simplification are listed below.

$$\begin{array}{cccc} Out(0,0) & Out(0,1) & Out(0,2) & Out(0,3) \\ Out(1,0) & Out(1,1) & Out(1,2) & Out(1,3) \\ \oplus\ Out(2,0) & Out(2,1) & Out(2,2) & Out(2,3) \\ Out(3,0) & Out(3,1) & Out(3,2) & Out(3,3) \\ \hline L'(0) & L'(1) & L'(2) & L'(3) \end{array}$$

Figure 2. Relation of the output of the Linear Predictor L'(j) to the output of a round of AES, Out(i,j).

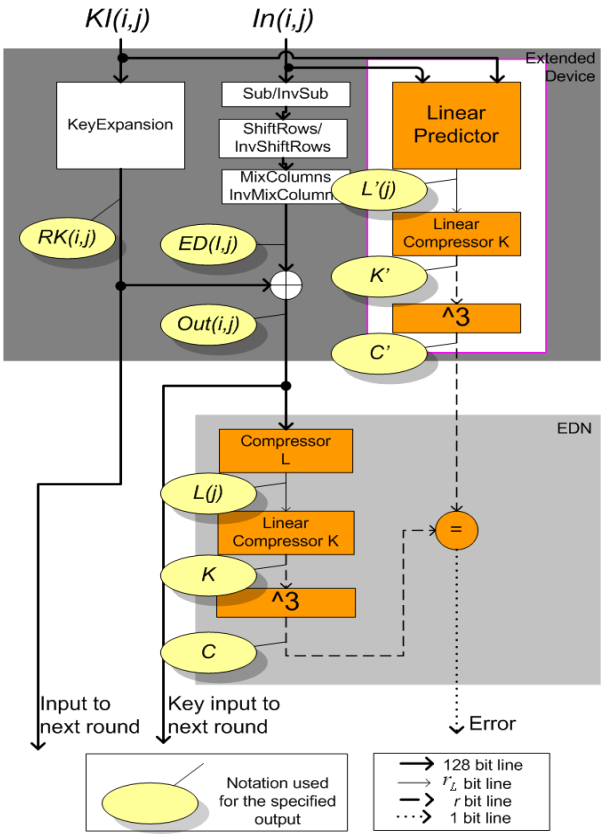

Figure 3. Notations used.

In the design presented in Figure 3 the complete linear predictor is actually the two components: Linear Predictor, and the Linear Compressor K.

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:

$$L'(j) = \bigoplus_{i=0}^{3} Out(i,j) = \bigoplus_{i=0}^{3} (RK(i,j) \oplus ED(i,j)) \quad \textbf{(7)}$$

where $j \in \{0,1,2,3\}$

If $L_{RK}'(j) = \bigoplus_{i=0}^{3} RK(i,j)$ and $L'_{ED}(j) = \bigoplus_{i=0}^{3} ED(i,j)$

where $j \in \{0,1,2,3\}$,

then $L'(j) = L_{RK}'(j) \oplus L_{ED}'(j)$ where $j \in \{0,1,2,3\}$

Thus, for the AES standard the following expression can be obtained for encryption:

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

where $\bullet$ is multiplication in $GF(2^8)$ and $Sub(In(i,j))$ is the SubBytes transformation on the byte $In(i,j)$ as defined in the AES standard [16].

Since:

$Sub(In(i,j)) = M(In(i,j)^{-1}) \oplus c$,

where $M = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$, $c = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

and inverse is in GF(2^8).

This further simplifies $L_{ED}'(0)$:

$$L_{ED}'(0) = M(In(0,0)^{-1} \oplus In(1,1)^{-1} \oplus In(2,2)^{-1} \oplus In(3,3)^{-1}) \quad \textbf{(8)}$$

Similarly,

$$\begin{aligned} L'_{RK}(0) = {} & M(KI(0,3)^{-1} \oplus KI(1,3)^{-1} \oplus KI(2,3)^{-1} \oplus \\ & KI(3,3)^{-1}) \oplus KI(0,0) \oplus KI(1,0) \oplus \\ & KI(2,0) \oplus KI(3,0) \oplus Rcon[i] \end{aligned} \quad \textbf{(9)}$$

where *Rcon[i]* is the Round Constant used in Key Expansion as defined in [16].

Combining **(7), (8)** and **(9)** results in:

$$\begin{aligned} L'(0) = {} & M(In(0,0)^{-1} \oplus In(1,1)^{-1} \oplus In(2,2)^{-1} \oplus In(3,3)^{-1}) \\ & \oplus M(KI(0,3)^{-1} \oplus KI(1,3)^{-1} \oplus KI(2,3)^{-1} \oplus KI(3,3)^{-1}) \\ & \oplus KI(0,0) \oplus KI(1,0) \oplus KI(2,0) \oplus KI(3,0) \oplus Rcon[i] \end{aligned}$$

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

$$\begin{aligned} L_{ED}'(1) &= M(In(0,1)^{-1} \oplus In(1,2)^{-1} \oplus In(2,3)^{-1} \oplus In(3,0)^{-1}) \\ L_{ED}'(2) &= M(In(0,2)^{-1} \oplus In(1,3)^{-1} \oplus In(2,0)^{-1} \oplus In(3,1)^{-1}) \\ L_{ED}'(3) &= M(In(0,3)^{-1} \oplus In(1,0)^{-1} \oplus In(2,1)^{-1} \oplus In(3,2)^{-1}) \\ L_{RK}'(1) &= L_{RK}'(0) \oplus KI(0,1) \oplus KI(1,1) \oplus KI(2,1) \oplus KI(3,1) \\ L_{RK}'(2) &= L_{RK}'(1) \oplus KI(0,2) \oplus KI(1,2) \oplus KI(2,2) \oplus KI(3,2) \\ L_{RK}'(3) &= L_{RK}'(2) \oplus KI(0,3) \oplus KI(1,3) \oplus KI(2,3) \oplus KI(3,3) \end{aligned}$$

Similarly for decryption:

$$\begin{aligned} L_{ED}'(0) = {} & (Minv(In(0,0) \oplus c))^{-1} \oplus (Minv(In(1,3) \oplus c))^{-1} \oplus \\ & (Minv(In(2,2) \oplus c))^{-1} \oplus (Minv(In(3,1) \oplus c))^{-1} \\ L_{ED}'(1) = {} & (Minv(In(0,1) \oplus c))^{-1} \oplus (Minv(In(1,0) \oplus c))^{-1} \oplus \\ & (Minv(In(2,3) \oplus c))^{-1} \oplus (Minv(In(3,2) \oplus c))^{-1} \\ L_{ED}'(2) = {} & (Minv(In(0,2) \oplus c))^{-1} \oplus (Minv(In(1,1) \oplus c))^{-1} \oplus \\ & (Minv(In(2,0) \oplus c))^{-1} \oplus (Minv(In(3,3) \oplus c))^{-1} \\ L_{ED}'(3) = {} & (Minv(In(0,3) \oplus c))^{-1} \oplus (Minv(In(1,2) \oplus c))^{-1} \oplus \\ & (Minv(In(2,1) \oplus c))^{-1} \oplus (Minv(In(3,0) \oplus c))^{-1} \\ L_{RK}'(0) = {} & M((KI(0,2) \oplus KI(0,3))^{-1} \oplus (KI(1,2) \oplus KI(1,3))^{-1} \oplus \\ & (KI(2,2) \oplus KI(2,3))^{-1} \oplus (KI(3,2) \oplus KI(3,3))^{-1}) \oplus \\ & Rcon[i] \\ L_{RK}'(1) = {} & KI(0,0) \oplus KI(1,0) \oplus KI(2,0) \oplus KI(3,0) \oplus KI(0,1) \oplus \\ & KI(1,1) \oplus KI(2,1) \oplus KI(3,1) \\ L_{RK}'(2) = {} & KI(0,1) \oplus KI(1,1) \oplus KI(2,1) \oplus KI(3,1) \oplus KI(0,2) \oplus \\ & KI(1,2) \oplus KI(2,2) \oplus KI(3,2) \\ L_{RK}'(3) = {} & KI(0,2) \oplus KI(1,2) \oplus KI(2,2) \oplus KI(3,2) \oplus KI(0,3) \oplus \\ & KI(1,3) \oplus KI(2,3) \oplus KI(3,3) \end{aligned}$$

where *Minv* is the inverse in $GF(2)$ of the matrix M defined above.

If the redundancy, and hence the size r of the cubic signature, is chosen such that it is smaller or equal than the output of the linear predictor r_L ($r_L \leq 32$), then the output of the linear predictor has to be first compressed before it is cubed. In the proposed design this is the role of the Compressor K. This compressor could be implementing multiplication over GF(2) by any (r_L *x* r) matrix with rank r.

The above design results in a linear predictor which protects the encryptor, decryptor and key expansion with

TABLE 2. FPGA implementations of robust cubic AES

SIZE OF THE CUBIC SIGNATURE *r*	PRIMITIVE POLYNOMIAL	CUBE SIZE (SLICES)	TOTAL SIZE (SLICES)	AREAD OVERHEAD (%)	FREQ. (MHz)	THROUGHPUT (Mb/s)	SPEED OVERHEAD (%)	PROB. OF AN UNDETECTABLE ERROR *
0	-	-	2,253	0	19.67	228	-	
8	$x^8+x^4+x^3+x+1$	28	3,362	49	15.92	185	19	2^{-16}
16	$x^{16}+x^5+x^3+x^2+1$	150	3,595	59	15.41	179	21	2^{-32}
20	$x^{20}+x^{17}+1$	202	3,683	63	15.21	177	22	2^{-40}
24	$x^{24}+x^7+x^2+x+1$	368	4,015	78	14.91	173	24	2^{-48}
28	$x^{28}+x^3+1$	349	3,996	77	17.02	198	13	2^{-56}
29	$x^{29}+x^2+1$	359	4,000	77	16.13	187	18	2^{-58}
31	$x^{31}+x^3+1$	452	4,133	83	16.76	195	14	2^{-62}
32	$x^{32}+x^{22}+x^2+x+1$	747	4,756	111	15.72	182	20	2^{-64}

* Estimated, based on Table 1. Assumes all errors at the extended output are equiprobable.

only about a 50% hardware overhead in a FPGA implementation when compared to the unprotected design.

6. FPGA implementations

We have implemented AES-128 (AES with 128-bit key) on a Xilinx XCV1000E FPGA using Xilinx Foundation 5.2 tools. The design included encryption/decryption capability with dynamic key expansion for both encryption and decryption. A simple control unit was implemented to control the whole circuit. That design was then protected by the proposed robust codes with different lengths of cubic signatures *r*. The required overheads are summarized in Table 2. Table 2 represents the relative size and speed of the circuit in post synthesis estimations only (not after place and route).

From the cube size column in Table 2 it is evident that the complexity of the network computing cubic signatures does not have uniform $O(r^2)$ growth behavior (even though the sizes are presented in slices, the same characteristic would still be evident for the corresponding gate-counts). In some instances it is absent such as in the cubes where *r*=24 and *r*=28. In that case, the larger *r* corresponds to a smaller complexity of the cubic network. This abnormality is a result of the selection of primitive polynomials of degree r defining GF(2^r) for each *r*. The size and the complexity of the cubic network are largely dependent on the characteristics of the primitive polynomial. In general, less terms in the polynomial are and smaller the degree of the non-leading term the better. These polynomial characteristics also explain the large cost of the 32-bit cubic network. The best primitive polynomial for the 32-bit case is relatively bad since it has 5 terms and it has a large order term (x^{22}).

Based on the results in Table II, we had chosen an *r*=28 for out test FPGA design. We felt that the *r*=28 case provided the best area to error protection compromise.

Table 3. Relative sizes of components for robust cubic architectures.

	CONTROL	AES ENC/DEC KEY EXPANSION CORE	LINEAR PREDICTOR	28 BIT CUBE	EDN WITH 28 BIT CUBE
SIZE (SLICES)	31	2,201	1,000	349	404
RELATIVE SIZE %	0.7 %	55.3 %	25 %	8.8 %	10.2 %

Table 3 represents the relative size of each component in the design where *r=28*.

7. Simulation results

The error coverage of the robust protection scheme with *r*=28 was simulated and compared to the error coverage of the corresponding linear protection scheme with the same redundancy. The 28-bit signature was then calculated from the 128-bit random input. In the robust simulation the signature was additionally cubed to produce the 156-bit extended output of the device (see Figure 4). It was in the extended 156-bit output that an error was injected. The same random inputs and the same error patterns were injected into the linear and robust cubic architectures. For each error pattern injected, five random inputs were simulated.

The results for random uniformly distributed symmetrical error pattern (componentwise XOR) injection are presented in Table 4.

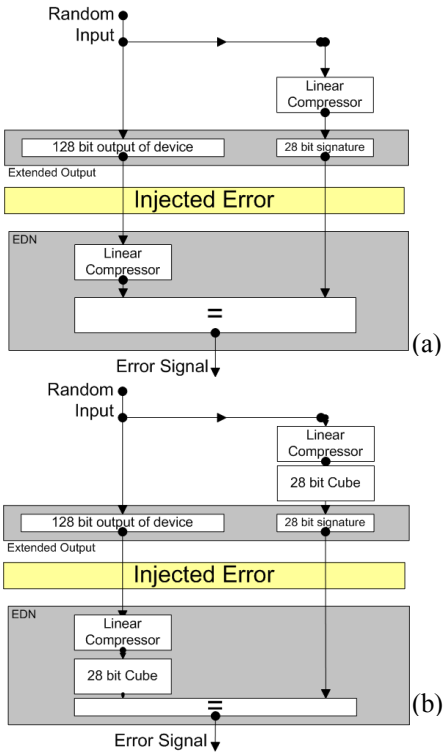

Figure 4 . Bit flip error model simulation test benches (a) linear; (b) robust cubic.

Table 4. Results of the injection 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

Results in Table 4 support our calculations (see Table 1). As expected, for symmetrical errors the probability of an undetectable error in the linear error detection scheme is 2^{-r} and much less in the robust case (calculated to be 2^{-2r}, hence no missed errors).

Similarly, random unidirectional (1 to 0 only) errors were injected at the extended output. The results of the simulation are presented in Table 5.

Table 5. Results of the injection of unidirectional errors

	ERROR PATTERNS INJECTED	ERRORS MISSED AFTER 1 RANDOM INPUT	ERRORS MISSED AFTER 2 RANDOM INPUTS
LINEAR PROTECTION	12.2 billion	160	0
ROBUST CUBIC PROTECTION	12.2 billion	43	0

The manifestation of unidirectional errors is data dependant. Thus, even the linear code exhibited some robust behavior. The data dependant error detection property of the linear code in the case of unidirectional errors caused the linear protection to be more closely matched to the robust protection (Table 5).

Errors, which were missed by the linear architecture, were also considered. In this simulation random 128-bit errors in information bits were generated. The 28 additional redundant error pattern bits were generated in such a way that the error pattern itself was a codeword of the corresponding (156,128) linear code. The results of the simulation are presented in Table 6.

The simulation results presented in Tables 4, 5 and 6 support the estimations for robust codes presented in Section 2. The simulation results support the claim that the number of undetectable errors for robust codes is considerably lower than that of the linear counterparts.

Table 6. Protection against errors missed by linear architecture

	ERROR PATTERNS INJECTED	ERRORS MISSED AFTER 1 RANDOM INPUT	ERRORS MISSED AFTER 5 RANDOM INPUTS
LINEAR PROTECTION	28.1 billion	28.1 billion	28.1 billion
ROBUST CUBIC PROTECTION	28.1 billion	157	84

8. Advantages of proposed robust architecture and future tasks

When attempting to protect a device against naturally occurring faults, assumptions and statistical analyses are made to determine the most probable errors. With natural errors in devices and communication channels there are often well-defined classes of errors, which appear the vast majority of the time. Protection schemes are then focused on protecting against that class of errors. However, statistical assumptions cannot, and should not be made for devices which can be subjected to an organized fault analysis attack. An attacker could potentially inject any type or kind of fault. An approach should be taken to minimize the number of total undetectable errors for such a device as to limit an attacker's chances of success.

The probability of injecting an undetectable error into a device is an important criterion to characterize a resistance to DFA attacks. The proposed robust codes can be used to extend the error coverage of existing linear codes. They have a hardware cost, but their increased error coverage advantage outweighs those costs when it comes to DFA resistant AES applications such as smart cards. As summarized in Table I, the number of

undetectable errors for the robust code is 2^r times smaller than for the corresponding linear code. When the redundancy is as large as in our example where r=28, the difference is enormous.

In our robust design we had chosen an r=28. With the introduction of the nonlinear cubic we reduced the fraction of undetectable errors from 2^{-28} to 2^{-56} without increasing the redundancy r of the original linear code and with relatively small hardware overhead (75% over unprotected AES). To achieve the same probability of 2^{-56} with linear codes only, the redundancy r would need to be extended to 56 bits. A linear predictor which can generate r=56 bit signatures would result in duplication of hardware. Thus, the robust nonlinear codes allow larger error coverage at a lower cost.

It is become clear now that tamper-resistance must be integral and a countermeasure against one physical attack must not benefit another attack [20]. The AES algorithm does not contain arithmetic operations, only table lookups, bitwise "XOR", and fixed byte rotation. Thus, AES is not particularly susceptible to leaking information during cryptographic operations through power and electromagnetic side-channels. Of course, now that new more effective attacks (e.g. [8, 21]) have been developed it is become clear that the algorithm itself can't be safe enough against all possible attacks and any reliable implementation must be done with reasonable countermeasures to side-channel attacks. However, it is still very important to preserve good properties of AES related to resistance against power and timing analysis attacks when one is suggesting a countermeasure to fault injection based attacks. Error detection procedures should be based on the same finite field operations as the AES algorithm, the proposed robust protection satisfy this criterion.

Our future research will combine protection against fault-injection with countermeasures against power and timing analysis attacks. There are two relatively new approaches that are fighting with the sources of data-dependent variations of power, timing and erroneous behavior [22, 23]. The first makes power consumption almost data-independent by balancing the capacitive loads of differential nodes. The second uses self-timed circuits with dual-rail logic to resist power analysis and single-point fault induction. These approaches (combined with methods proposed in [23] and [22]) seem to be very attractive countermeasure against power-analysis attacks. Self-timed circuitry because of absence of clocks makes glitch attack practically impossible. However, dual-rail encoding with alarm state from [23] is effective only for a limited class of induced errors (a single point fault induction, i.e. affecting only one wire). It is rather easy to imagine an attacker that can modify symmetrically both wires from dual-rail pair using e.g. attack from [17,18]. Such error induction will not be visible for the alarm generators from [23]. More developed error detection techniques, such as robust protection, need to be used. Other weaknesses of "classical" self-timed implementations [23] discovered in [18] could be effectively dealt with by asynchronous fine-grain multidimensional pipelined structures [24]. We are now starting development of robust asynchronous fine-grain pipelined implementations of AES using EDA tools under development in Boston University.

References:

[1] C.E. Landwehr, *Computer Security*. International Journal of Information Security (2001) 1: 3-13.

[2] E. Hess, N. Janssen, B. Meyer, and T. Schütze *Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures - A Survey*. Proceedings of EUROSMART Security Conference, 2000

[3] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, *Side Channel Cryptanalysis of Product Ciphers*, ESORICS '98 Proceedings, 1998, pp. 97-110.

[4] C. D. Walter, *Montgomery's Multiplication Technique: How to make it Smaller and Faster*, Proc. Workshop on Cryptographic Hardware and Embedded Systems, (CHES 99), 1999, Lecture Notes in Computer Science, vol. 1717, pp 80-93.

[5] M.Joye, J.-J.Quisquater, S.-M. Yen and M.*Yung Observability Analysis - Detecting when Improved Cryptosystems Fail*. Topic in Cryptology –CT-RSA 2002, vol. 2271 in Lecture Notes in Computer Science, pp 17-29.

[6] S. Micali and L. Reyzin, *Physically Observable Cryptography*, Cryptology ePrint Archive of IACR, No. 120, 2003, available at http://eprint.iacr.org/2003/120

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

[8] C.N. Chen and S.-M.Yen, *Differential Fault Analysis on AES Key Schedule and Some Countermeasures*, ACISP 2003, LNCS 2727, pp.118-129, 2003

[9] 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

[10]C. Giraud. *DFA on AES.* Cryptology ePrint Archive, Report 2003/008. Available: http://eprint.iacr.org and http://citeseer.nj.nec.com/558158.html

[11]J. Blomer and J.P. Seifert, *Fault based cryptanalysis of the advanced encryption standard (AES),* Cryptology ePrint Archive: Report 2002/075. Available at: http://eprint.iacr.org.

[12]Ramesh Karri, Kaijie Wu, Piyush Mishra, Yongkook Kim, Concurrent Error Detection of Fault Based Side-Channel Cryptanalysis of 128-Bit Symmetric Block Ciphers. *IEEE Transactions on COMPUTER-AIDED DESIGN of Integrated Circuits and Systems,* Vol.21, No.12, pp. 1509-1517, 2002

[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]M. G. Karpovsky, P. Nagvajara, "Optimal Robust

Compression of Test Responses," *IEEE Trans. on Computers*, Vol. 39, No. 1, pp. 138-141, January 1990.
[15]M. G. Karpovsky, P. Nagvajara, "Optimal Codes for the Minimax Criterion on Error Detection," *IEEE Trans. on Information Theory*, November 1989.
[16]FIPS PUB 197: *Advanced Encryption Standard,* http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[17]S. Skorobogatov and R. Anderson. *Optical Fault Induction Attacks*. IEEE Symposium on Security and Privacy, May 2002.
[18]J.J.A. Fournier, S. Moore, H.Li, R. Mullins, and G. Taylor. *Security Evaluation of Asynchronous Circuits*. Proc. Workshop on Cryptographic Hardware and Embedded Systems, (CHES 2003).
[19] F.J. McWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes,* North Holland, 1978
[20]S.-M.Yen, S.Kim, S.Lim and S.*Moon A Countermeasure against One Physical Cryptanalysis May Benefit Another Attack*: ICICS 2001, LNCS 2288, pp. 414-427
[21]S.-M.Yen *Amplified differential power cryptanalysis of some enhanced Rijndael implementations*. ACISP 2003, LNCS 2727, pp.106-117, 2003
[22]Kris Tiri, Moonmoon Akmal, Ingrid Verbauwhede, "*A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards*", 28th European Solid-State Circuits Conference (ESSCIRC 2002)
[23]Simon Moore, Ross Anderson, Robert Mullins, George Taylor, Jacques Fournier, *Balanced Self-Checking Asynchronous Logic for Smart Card Applications* Microprocessors and Microsystems, 27 (2003) pp. 421-430.
[24]A.Taubin, K. Fant, J. McCardle *Design of Delay-Insensitive Three Dimension Pipeline Array Multiplier for Image Processing*. Proceedings, 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD'2002, p.p.104-111.