Parity-Based Concurrent Error Detection in Symmetric Block Ciphers

Ramesh Karri
Electrical and Computer Engineering Dept
Polytechnic University
6 Metrotech Center
Brooklyn, NY 11201

Grigori Kuznetsov and Michael Goessel
University of Potsdam
Institute of Computer Science,
Fault Tolerant Computing Group
D-14439 Potsdam, Germany

Abstract

Deliberate injection of faults into cryptographic devices is an effective cryptanalysis technique against symmetric and asymmetric encryption. We will describe a general concurrent error detection (CED) approach against such attacks on symmetric block ciphers using CS-cipher as an example. The proposed CED compares a carefully modified parity of the input plain text with that of the output cipher text.

An analysis of the CS-Cipher shows that on one hand the parity of its inputs is modified by a constant one or zero by component-wise exclusive-or of inputs with the round keys and with the round constants; if the parity of the round keys and of the round constants is odd (even) the parity of the inputs is modified by one (not modified). On the other hand, the diffusion network based on the Fast Fourier Transform does not alter the parity. Finally, the 16-bit to 16-bit non-linear mixing function does not have any simple relation between the parity of its inputs and of its outputs. The mixing function is composed of a linear function φ and a non-linear function p. In order to maintain the invariance of the parity from the inputs to the outputs of each CS-cipher encryption round, we added a parity correction term (exclusive-or of the parity of its inputs and outputs) to the non-linear function p and slightly modified the parity function due to the linear function φ. Faults introduced into the CS-cipher design are detected by comparing the overall parity of the input modified by the parity of the round keys, round constants and correction terms of the mixing functions with the parity of the (intermediate) cipher text obtained after every step of an encryption round.

1. Introduction

Until recently cryptanalysts analyzed cipher systems by modeling them as ideal mathematical objects and applying differential cryptanalysis [5] and linear cryptanalysis [17]. Although these techniques are useful in exploring weaknesses in algorithms, they do not exploit weaknesses in their implementations. Hardware and Software implementations of (crypto) algorithms leak information via side-channels such as time consumed or power dissipated by the operations used, electromagnetic radiation emitted by the device and faulty computations resulting from deliberate introduction of faults. Differential and linear cryptanalysis techniques can be combined with such side-channel information to break the secret key and/or the implementation details of the cipher. Kelsey et.al., [14] showed that even a small amount of side-channel information is sufficient to break common ciphers. For example, Differential Fault Analysis (DFA) exploits deliberate introduction of faults into the system and requires between 50 to 200 cipher text blocks to recover a key of symmetric block cipher Data Encryption Standard (DES); the best non-side-channel attack requires approximately 64 terabytes of plain text and cipher text encrypted under a single key.

Side-channel attacks are a serious threat since they can be applied against a wide variety of applications; examples include intellectual proprietary rights violations (for example, reverse engineering pay-TV smart cards, pre-pay meter tokens, remote locking devices for cars or SIM cards for GSM mobile phones [1]) and extraction of the secret information (such as the pin number, health records, and bank history) stored in a small smart cards. Timing attacks exploit timing characteristics of the implementation of operations used in a cipher to break it [15]. Differential Power Analysis (DPA) attacks [16] divide the encryption time into a number of time slots. Then one power measurement is taken in each slot for different input plain texts. A small number of these measurements will correlate with each bit of internal stage during encryption. This attack does not require much knowledge of the implementation and yields both key and enough information to derive a model of the device. Side-channel attacks can be thwarted by carefully designing the software/hardware to either reduce the amount of side-channel information that leaks (for example, by using shielding to reduce electromagnetic radiation), or make the leakage irrelevant (for example, by performing redundant, random computations). Eliminating side-channel information or preventing it from being used to attack a secure system is an active area of research.

In this paper we present a parity-based approach to CED in Symmetric Block Ciphers. We will review the fault injection attacks, associated fault models and existing counter measures in section 2. In section 3 we will describe the CS-Cipher, in section 4 we will present our parity-based approach to CED and in section 5 we will evaluate the capability of the proposed CED method. Section 6 concludes the paper.

2. Fault injection attacks

Fault injection attacks are based on the observation that faults deliberately introduced into a crypto-device leak information about the implemented algorithms. These attacks are practical since elevated levels of radiation or heat, incorrect voltage, or atypical clock rate can cause a tamperproof device to malfunction. Boneh et. al., [7] presented the first fault injection attack. More recently, Biham and Shamir presented a fault injection attack of DES symmetric block cipher [6]. They presented a transient fault based Differential Fault Analysis (DFA) attack and a permanent fault based non-DFA attack to recover the last round key by using less than 200 cipher texts. Biham and Shamir extended their fault model to show that DFA can uncover the structure of an unknown cryptosystem implemented in an EEPROM based smart card. This attack is based on the observation that it is much easier to induce a $1\rightarrow0$ bit flip than to induce a $0\rightarrow1$ bit flip in an EEPROM. This attack used DES as the unknown cipher and required only about 500 faulty cipher texts to identify the bits in the right half, up to 5000 faulty cipher texts to identify the non-linear substitution box (s-box) operations and their input and output bits, and about 10000 faulty cipher texts to reconstruct all DES s-boxes.

Anderson and Kuhn described fault injection attacks targeting smartcard implementations [2]. In one of the attacks, they assumed that the instruction memory of smart cards can be corrupted. If in a process loop, the variable controlling the number of rounds is set to 1, encryption executes just one round, thereby compromising the round key. In a second attack they exploited the chip writing ability of the attacker. Assuming that the attacker is familiar with the implementation, they showed how the keys can be extracted from the card by overwriting specific memory locations.

2.1. Fault injection attacks: The Fault Models

Boneh et. al., [7] used a practical fault model wherein a fault is induced at a random bit location in one of the registers at some random intermediate round of a cryptographic computation. Biham and Shamir [6] use a similar realistic fault model wherein either a transient or a permanent fault is induced randomly into the device. They then adapt this basic fault model to factor in the asymmetric property of EEPROMs: it is much easier to induce a $1\rightarrow0$ bit flip than to induce a $0\rightarrow1$ bit flip. Anderson and Kuhn [2] used two different fault models for microcontroller based smartcards: in the first they assumed that the instruction memory of smart cards can be randomly corrupted and in the second they assumed that the attacker has the ability to write into specified locations in the memory. These and other fault attacks and associated fault models are summarized in [23, 24]. The proposed CED approach is applicable to the practical fault models described.

2.2. Fault injection attacks: CED

CED followed by suppression of the corresponding faulty output can thwart fault injection attacks on symmetric block ciphers; on detecting a faulty computation, the stored key is protected by suppressing the corresponding faulty cipher text.

Straightforward duplication and comparison of encryption and decryption hardware yields more than 100% hardware overhead. Alternatively, a spare module for each type can be used to detect faults in hardware modules of that type. Such a spares-based approach has been adopted in a hardware implementation of the 128-bit symmetric block cipher IDEA [8]. Spares-based approaches are suitable for block ciphers that use arithmetic operators, such as IDEA and RC6. Although hardware is not duplicated, an extra module for each operation type entails considerable hardware overhead, especially for encryption algorithms like Advanced Encryption Standard (AES) [10] and DES that use random, non-arithmetic operations such as S-Boxes.

Time redundancy based CED approaches involve encrypting (decrypting) the data a second time followed by the comparison of two results. Wolter et. al. [20] developed a CED technique for symmetric block cipher IDEA wherein the test data was encrypted and then decrypted. This approach entails more than 100% time overhead. Further, it can only tolerate transient faults if the data traverses identical paths through the encryption and decryption data paths both during the normal computation and during the re-computation.

A third CED approach involves encoding the message before encryption and checking it for errors after decryption. Wolter et. al. [20] used residue codes for fault detection in adders, multipliers, and exclusive-or units. Area overhead of this approach is due to the encoders at the input and decoders at the output to translate the plain and cipher texts into the internal code words. Another encoding scheme sets several bits of the message to a particular fixed value, 0 or 1 [11]. A mismatch between these fixed bits of deciphered text and the original plain

text detects an error. This scheme results in significantly less area overhead when compared to other encoding schemes, but has large fault detection latency and performance penalty since it uses some of the bits in messages for error detection. This scheme detects faults in the transmission channel; encrypted data is transmitted and fault is detected after decryption at the receiver.

In [9] a CED technique that predicted the inverse of the parity of the outputs from the parity of the inputs was proposed for the non-linear s-box and other functions used in DES. A similar technique that predicted the parity of the outputs from the parity of the inputs for the non-linear s-box and linear mixing functions used in the AES was proposed in [3]. They use one parity bit per byte for CED yielding a 256x9 bit modified s-box. In order to detect additional errors in the s-box (input parity errors and internal memory errors) they double the size of the s-box by proposing a 512x9 bit implementation yielding an area overhead of over 100% for s-box CED.

Karri et. al [13] developed a register transfer level CED approach for symmetric block ciphers by exploiting the inverse relationship between encryption and decryption at the algorithm level, round level and individual operation level. They demonstrated this inverse-relationship principle on 128-bit symmetric block ciphers including AES and AES candidates RC6 and Serpent. This approach assumes that the cipher device operates in a half-duplex mode (i.e. either the encryption or the decryption but not both are simultaneously active). Bertoni et. al. [4] applied this inverse-relationship principle to round key generation of the AES encryption algorithm.

3. CS-Cipher Symmetric Block Cipher

The architecture of a symmetric block cipher contains a key expansion module, an encryption module, and a decryption module. Key expansion module expands the user key into round keys and loads them into the key RAM prior to encryption or decryption. Using the round keys, the device encrypts (decrypts) the plain (cipher) text to generate the cipher (plain) text. Symmetric block ciphers have an iterative looping structure with each round using several operations and round key(s) to process the input data.

One encryption round and one decryption round of 64-bit CS-Cipher with the nine processing steps of a round are shown in Figure 1. The 64-bit input $x^i = (x^i_{63}\ldots x^i_0)$ of the i^{th} round is divided into four 16-bit groups $(x^i_{15}\ldots x^i_0)$, $(x^i_{31}\ldots x^i_{16})$, $(x^i_{47}\ldots x^i_{32})$ and $(x^i_{63}\ldots x^i_{48})$. Each 16-bit group consists of an 8-bit left half and an 8-bit right half. In the first step of the i^{th} round, the 64-bit round key k_i is exclusive-ored with the 64-bit input to the round. Steps 4 and 7 are similar to step 1 except that constants c and c' are used.

In the steps 2, 5 and 8, four 16-bit groups are processed by four instances of the mixing function M. In steps 3, 6 and 9 the inputs are byte-wise permuted using the Fast Fourier Transform (FFT) network Specifically, the FFT network permutes the input bytes 8 7 6 5 4 3 2 1 into output bytes 8 4 7 3 6 2 5 1.

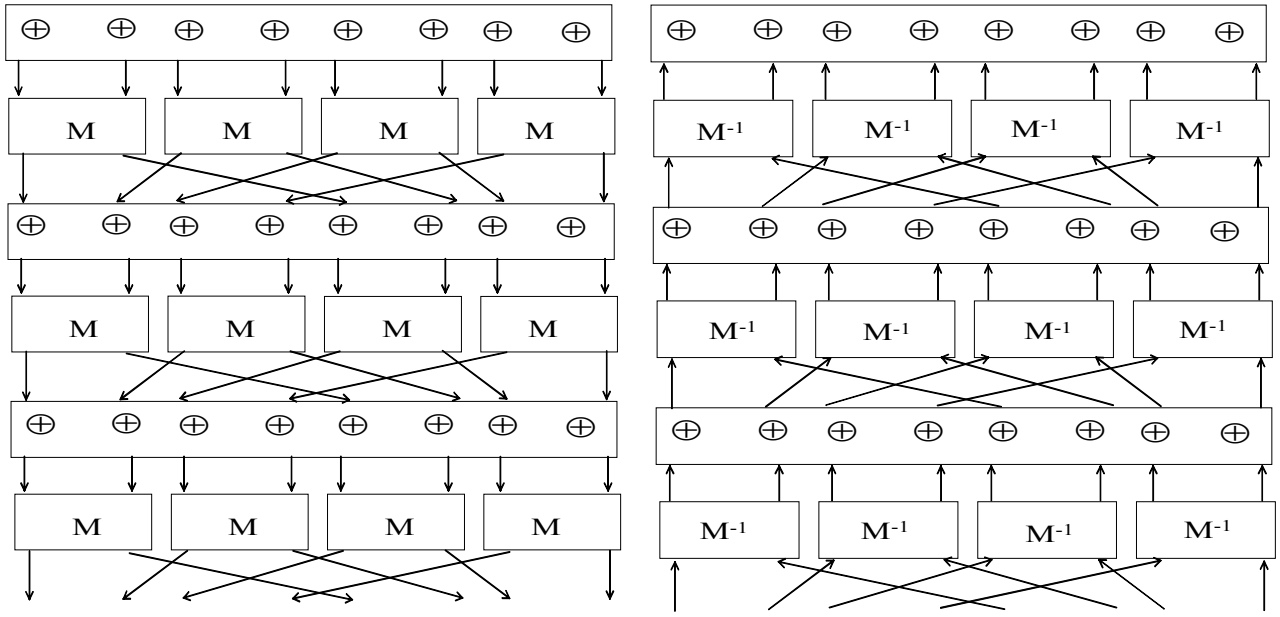

Figure 1: 64-bit CS-Cipher: (a) one round of encryption and (b) one round of decryption.

4. Concurrent Error Detection in CS-Cipher

In this section we propose a new method for concurrent error detection (CED) for the symmetric CS-cipher encryption algorithm. Protecting the CS-Cipher from fault injection attacks entails protecting the encryption/ decryption data paths as well as the RAM used to store the round keys. Since significant work has been done to protect RAMs and other memory structures using Parity code, Hamming code etc, we do not address CED for key RAM. Rather, we will focus on CED for encryption and decryption data paths.

In every round of the encryption algorithm we modify the parity of the inputs of this round in accordance with the actual processing steps into the parity of the outputs of this round and we compare this modified parity with the actual parity of the outputs of this round.

Now we will describe in detail the proposed method of CED for the first and the second processing steps of CS-Cipher encryption in a considered round. For CED we divide the inputs of the rounds into four groups of 16 bits each according to the word length of inputs of mixing function M.

For every group of 16 bits we determine the parity of the inputs of the considered group and modify this input parity according to the component-wise exclusive-or with the corresponding 16 bits of the key k and of the processing by the mixing function M into the parity of the right byte of the output of M. An error is detected by comparing the actual parity of the right byte of the outputs of the function M with the modified parity of the corresponding 16-bit input.

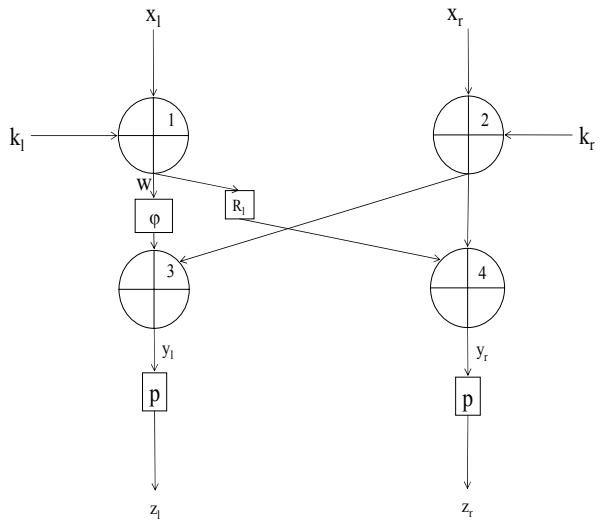

Figure 2: 16-bit input 16-bit output mixing function.

The CED for the left byte of the outputs of the mixing function M is similar with the only difference that instead of the parity of all 16 inputs of the group the parity of a subset of 12 of these inputs is modified into the parity of the left byte of the outputs of the mixing function M. This difference in the CED circuitry for the right and the left bytes is due to the non symmetric structure of the mixing function M.

Figure 2 describes the considered steps of the i^{th} encryption round in more detail for the first group of the 16 least significant input bits $x = x_{15} \dots x_0$ with $x_l = x_{15} \dots x_8$ and $x_r = x_7 \dots x_0$. The corresponding 16 bits of the key $k = k_{15} \dots k_0$ with $k_l = k_{15} \dots k_8$ and $k_r = k_7 \dots k_0$ are component-wise exclusive-ored in the 8 exclusive or elements 1 and 2 respectively. To avoid too many subscripts in the notation the key of this (i^{th}) round is simply denoted by k. Also the inputs of this (i^{th}) round are denoted by $x_{15} \dots x_0$. For $w = w_{15}, w_{14},\dots, w_8$, the function φ of the mixing function M is defined as:
$\varphi(w)=(w_{14}w_{13}w_{12}w_{11}w_{10}w_9w_8w_{15} \wedge 01010101) \oplus w_{15}w_{14}w_{13}w_{12}w_{11}w_{10}w_9w_8=$
$w_{15},w_{13}\oplus w_{14},w_{13},w_{11}\oplus w_{12},w_{11},w_9\oplus w_{10},w_9,w_8\oplus w_{15}.$

Substituting $w= x_l\oplus k_l = x_{15}\oplus k_{15,}\ x_{14}\oplus k_{14,} \dots x_8\oplus k_{8,}$ we have

$\varphi(x_l\oplus k_l) = x_{15}\oplus k_{15,}\ x_{13}\oplus k_{13}\oplus x_{14}\oplus k_{14,}\ x_{13}\oplus k_{13,}\ x_{11}\oplus k_{11}\oplus x_{12}\oplus k_{12,}\ x_{11}\oplus k_{11,}\ x_9\oplus k_9\oplus x_{10}\oplus k_{10,}\ x_9\oplus k_{9,}\ x_8\oplus k_8\oplus x_{15}\oplus k_{15.}$

According to Figure 2 y_l is determined as:

$y_l = \varphi(x_l\oplus_8 k_l)\oplus_8 x_l\oplus_8 k_l$, we obtain for parity

$P(y_l)=y_{14}\oplus y_{13}\dots\oplus y_8=x_{14}\oplus x_{12}\oplus x_{10}\oplus x_8\oplus P(x_r)\oplus\gamma,$

with $P(x_r)= x_7 \oplus x_6 \oplus\dots\oplus x_0$

$\gamma = k_{14}\oplus k_{12}\oplus k_{10}\oplus k_8\oplus P(k_r)$ and

$P(k_r) = k_7\oplus k_6\dots\oplus k_{0.}$

Left rotate function $R_l(x_l\oplus k_l)$ is defined as:

$R_l(x_l\oplus k_l) = x_{14}\oplus k_{14,}\ x_{13}\oplus k_{13,}\ \dots x_8\oplus k_8, x_{15}\oplus k_{15}$ and for the parity of y_r

$P(y_r) =x_1\oplus k_1\oplus R_l(x_l\oplus k_l)$ we obtain

$P(y_r)=P(x)\oplus P(k)$ with

$P(x)=x_{15}\oplus x_{14}\oplus\dots\oplus x_0$ and $P(k)= k_{15}\oplus k_{14}\dots\oplus k_{0.}$

y_l and y_r are applied to a non-linear function p with left output byte $z_l = p(y_l)$and right output byte $z_r =p(y_r)$. The non-linear function p has been designed to satisfy properties such as high nonlinearity, low differential uniformity etc that have been shown to reflect its strength against linear and differential cryptanalysis [8].

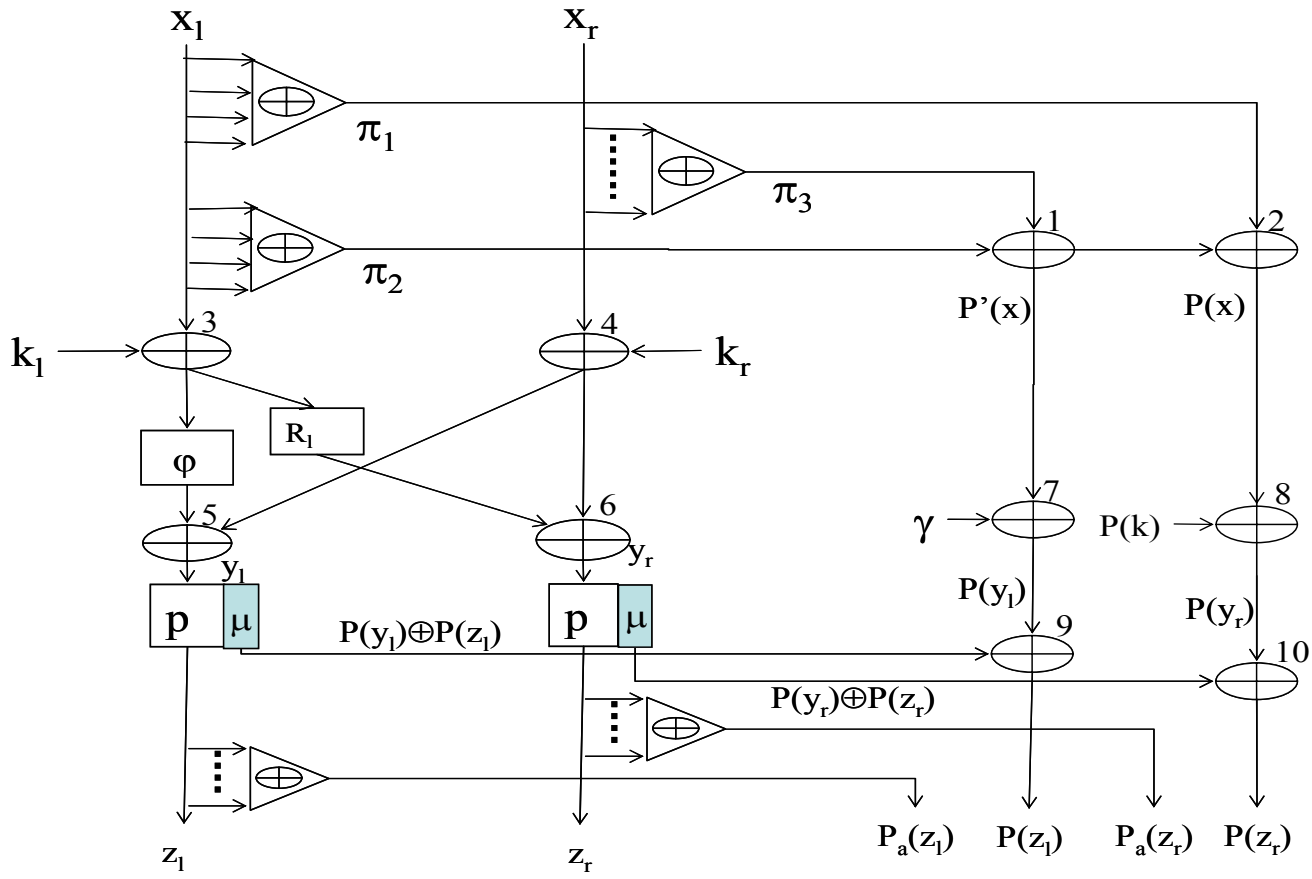

Figure 3: CED architecture for the mixing function M.

Based on this analysis of the first two steps of a CS-cipher encryption round we are now able to design the CED architecture for the exclusive or with the key bits operation and for the mixing function as shown in Figure 3. As we already pointed out the parity P(x) of the inputs is modified according to the processing steps into the parity $P(z_r)$ of the eight least significant outputs $z_r=z_7,\ldots,z_0$ and compared with the actual parity $P_a(z_r)$ of the outputs $z_7,\ldots,z_0$ of the right nonlinear p function.

Similarly the parity P'(x) of the inputs $x_{14},x_{12},x_{10},x_8,\ldots,x_0$ is modified into the parity $P(z_l)$ of the most significant outputs $z_{15},\ldots, z_8$ of the left non-linear p function. This modification will now be described in detail. The input parity P(x) is implemented as $P(x) = \pi_1\oplus\pi_2\oplus\pi_3$ where:

$\pi_1 = x_{15}\oplus x_{13}\oplus x_{11}\oplus x_9$,

$\pi_2 = x_{14}\oplus x_{12}\oplus x_{10}\oplus x_8$, and

$\pi_3 = P(x_r)= x_7 \oplus x_6 \oplus\ldots\oplus x_0$.

Since we have $P(y_r)=P(x)\oplus P(k)$, in a first step the input parity P(x) is modified by exclusive-oring P(x) with P(k). $P(k)\in\{0,1\}$ can be pre-computed for a given key k.

For non-linear function p, we add an additional output $\mu(y_r)$ implementing exclusive-or of the input parity and output parity.

$\mu(y_r)=y_7\oplus y_6\oplus\ldots\oplus y_0\oplus z_7(y_r)\oplus z_6(y_r)\oplus\ldots\oplus z_0(y_r)$

$= P(y_r)\oplus P(z_r)$

where $P(y_r)= y_7\oplus y_6\oplus\ldots\oplus y_0$ of the eight inputs of the function p and $P(z_r)= z_7(y_r)\oplus z_6(y_r)\oplus\ldots\oplus z_0(y_r)$ of the eight outputs of the p function. Using this transformation, $P(y_r)$ is modified into $P(y_r)\oplus\mu(y_r) = P(y_r)\oplus P(y_r)\oplus P(z_r) = P(z_r)$. The function μ can be implemented by a combinational circuit or if p is implemented in a ROM an additional output can be easily added.

The modified parity $P(z_r)$ of z_r is compared with its actual parity $P_a(z_r)$. $P_a(z_r)$ is generated by an exclusive or tree of the eight outputs of the right hand side p function in Figure 3.

The parity $P(y_l)$ is determined as

$P(y_l) = x_{14}\oplus x_{12}\oplus x_{10}\oplus x_8\oplus P(x_r)\oplus\gamma=\oplus\pi_2\oplus\pi_3\oplus\gamma$

$= P'(x)\oplus\gamma$, with

$\gamma = k_{14}\oplus k_{12}\oplus k_{10}\oplus k_8\oplus P(k_r)$.

Thus the parity $P'(x)=\pi_2 \oplus \pi_3$ is modified by performing an exclusive-or of $P'(x)$ with γ into the parity $P(y_l)$ of y_l. Similar to $P(k)$, γ can be pre-computed from the key k. As in the case of the outputs z_r we add a single additional output $\mu(y_l)$ implementing $\mu(y_l)=P(y_l)\oplus P(z_l)$ to the left hand side p function with eight inputs and eight outputs in Figure 3. $P(y_l)$ is modified into $P(y_l) \oplus \mu(y_l)=P(y_l)\oplus P(y_l)\oplus P(z_l)=P(z_l)$. This modified parity $P(z_l)$ is compared with the actual parity $P_a(z_l)$ determined from the eight outputs of the left p function in Figure 3.

In the following steps of the encryption round being considered the data are byte-wise FFT permuted as shown in Figure 1 (a). Instead of key k, a constant c is bit-wise exclusive or-ed and the data is once again processed by the mixing function as described previously. Finally, this step is repeated a third time using constant c' instead of c. The repetitive structure of bit-wise exclusive or with a constant, application of the mixing function and byte-wise FFT permutation allows us in a very simple way to implement CED into the CS-Cipher block cipher. Figure 4 shows such a CS-Cipher encryption round implementation that uses one level of exclusive (for steps 1, 4, and 7), four mixing functions (for steps 2, 5 and 8) and one FFT permutation (for steps 3, 6, and 9).

Each encryption round consumes three clock cycles. The exclusive-or units in Figure 4 exclusive-or the inputs with the round keys (or with the round constants, c and c'). The mixing functions in Figure 4 include the CED circuitry for their left and their right halves of their inputs as shown in Figure 3. In this CED architecture, additional parity trees are not necessary for computing the output parities of each mixing function. The output of a clock cycle (output of steps 3, 6, and 9) is the input to the next clock cycle (steps 1, 4 and 7). Hence the parity trees used to compute the parity for the inputs in steps 1, 4 and 7 can be used to compute the output parities of the mixing functions (in steps 2, 5 and 8) in the previous clock cycle in all rounds.

The output parity $P(z_l^1)$ of the left output of M_1 is computed by the left half of the parity tree at the input to the mixing function M_1 and the output parity $P(z_r^1)$ of the right output of M_1 is computed by the left half of the parity tree at the input to the mixing function M_3. Similarly, the output parity $P(z_l^2)$ of the left output of M_2 is computed by the right half of the parity tree at the input to the mixing function M_1 and the output parity $P(z_r^2)$ of the right output of M_2 is computed by the right half of the parity tree at the input to the mixing function M_3. For the outputs of mixing functions M_3 and M_4 we can proceed accordingly. As shown in Figure 3, this is possible by storing the results after each clock cycle in a round and after every round.

The parity of the round keys can be computed at the time of round key generation. Hence the overhead associated with it can be absorbed into the one time cost of round key generation. A more general approach of CED applied to substitution permutation networks is described in our patent application [21].

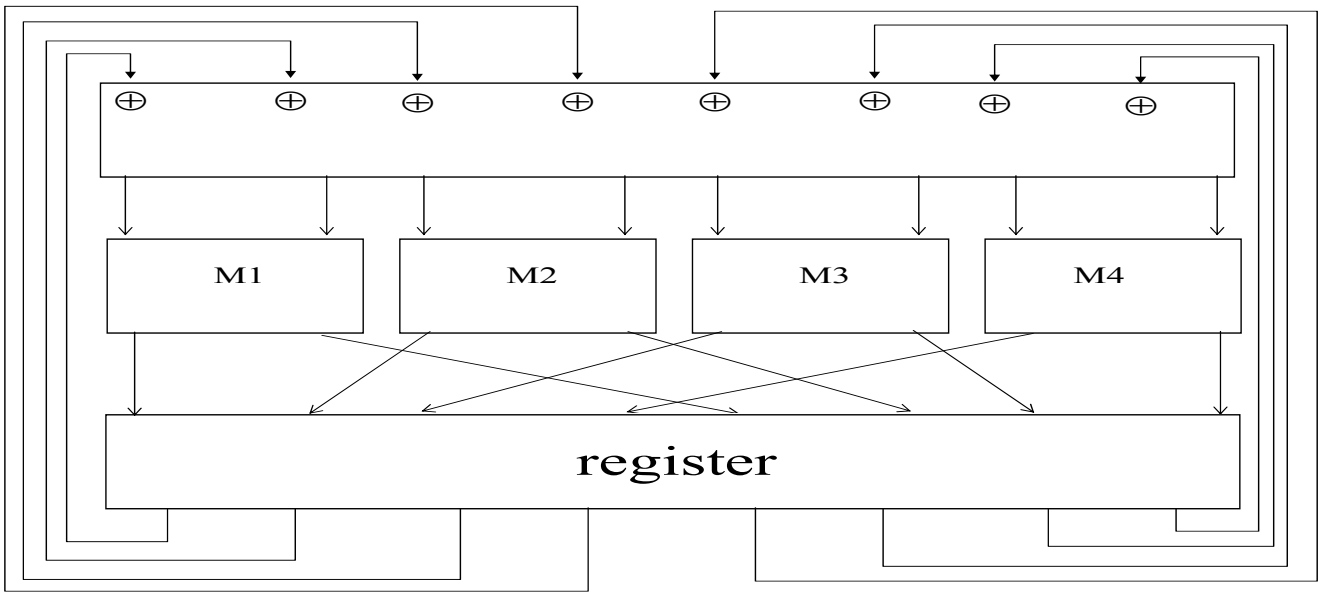

Figure 4: Architecture for CS-Cipher with CED

5. Error Detection Capability

In symmetric block ciphers a single-bit error (which may be caused at the outputs of a processing step by a fault in the hardware implementing this step) will typically result in a large number of erroneous bits after only a small number of processing steps. This is called the strict avalanche criterion [22]. One can expect that with a probability 0.5 the number of erroneous bits is odd. This limits the applicability of conventional parity prediction for CED in block ciphers; the probability of error detection by this method can also be expected to be 0.5. In contrast, the fast spread of a single-bit error into a large number of erroneous bits has no influence on probability of error detection of the proposed CED.

In the proposed CED, the parity of the inputs of every encryption round is modified step-by-step into the parity of the outputs of that round in accordance with the successive processing steps of the round. If a technical or a deliberately injected fault in the hardware of the i^{th} processing step results in an error in an odd number of output bits it is detectable by parity. The modified parity up to the i^{th} processing step is then different from the actual parity of the (erroneous) outputs of the i^{th} processing step. If the remaining processing steps in a round are fault-free then in these processing steps the parity of their (now erroneous) inputs is correctly modified into the parity of their outputs. Thus the difference between the parity of the input to the round modified up to the i^{th} step and the parity of the outputs of the i^{th} processing step of the round are detected by comparing the parity of the actual outputs of the round with modified parity of the inputs of this round (see [21]).This reduces error detection analysis of a round into error detection analysis for every processing step of a round.

Consider the error detection capability of the proposed CED method. According to Figure 3 the input parities P(x) and P'(x) are determined by the exclusive-or trees π_1, π_2, and π_3. Clearly, every single stuck-at fault in these exclusive-or trees, in the exclusive-or trees for computing $P_a(z_l)$ and $P_a(z_r)$ and in the exclusive-or elements 1,2,7,8,9 and 10 will be detected.

A single stuck-at fault in the exclusive-or elements 3, 4 and 6 will result in an erroneous value y_r^e instead of the correct output y_r with $P(y_r^e) \neq P(y_r)$. Then y_r^e is mapped into z_r^e by the p function (i.e. $z_r^e = p(y_r^e)$) and the parity is modified by $\mu(y_r^e) = P(y_r^e) \oplus P(z_r^e)$. Finally, $P(y_r) \oplus P(y_r^e) \oplus P(z_r^e)$ is compared with $P(z_r^e)$. Since we have $P(y_r) \neq P(y_r^e)$ the error is detected. A fault in the linear function ϕ with an odd number of erroneous bits results in an erroneous output y_l^e instead of the correct output y_l with $P(y_l^e) \neq P(y_l)$. Similarly, y_l^e is mapped by the function p into $z_l^e = p(y_l^e)$ and the modified input parity $P(y_l)$ is further modified by $\mu(y_l^e) = P(y_l^e) \oplus P(z_l^e)$ into $P(y_l) \oplus P(y_l^e) \oplus P(z_l^e)$. This modified parity is compared with the actual parity $P(z_l^e)$ of the outputs of p, and since we have $P(y_l) \neq P(y_l^e)$ the error is detected.

To detect faults in the p-function a special implementation of this function is considered. Every output of the p-function, including the additional output μ is separately implemented as a combinational circuit. Thereby no output shares a gate with another output. This special implementation guarantees that every single stuck-at fault in the implementation of the p-function results in a single bit error at its outputs which is detectable by parity. Thus we have shown that for errors due to single stuck-at faults and odd bit errors 100% error detection is achieved by the proposed method.

6. Conclusions

In this paper a new parity based method for concurrent error detection for symmetric block-ciphers was proposed. The parity of the inputs of an encryption round is modified step-by-step in accordance with the processing steps of the block cipher round into the parity of the outputs of the considered round. By comparing the actual parity of the outputs of the round with the modified parity of the inputs an error will be detected. In a special implementation of the CS-Cipher we showed that a single exclusive-or tree can be used to compute the parity of the inputs and outputs of round.

The processing steps of the CS-cipher consist of bit-wise exclusive-or of the data with round-key (or round-constant), byte-wise permutation and processing by a non-linear mixing function. Bitwise exclusive-or with a round key (or with a constant) changes the parity of the input by the parity of the key or the parity of the constant (and this parity can be determined during key computation). The byte-wise permutation does not alter the parity. The mixing function is composed of a linear function ϕ and a non-linear function p. To modify the parity of the inputs of the mixing function to the parity of their outputs a one-bit correction term μ (that implements the exclusive-or sum of the parity of the inputs and the outputs) is added. Finally, the parity function is slightly modified in accordance with the linear function ϕ.

If all outputs of the p-function are separately implemented all errors due to single stuck at faults and injected odd bit errors are detected. This result is independent of the fact that symmetric block ciphers single bit errors rapidly spread out into multiple-bit errors due to the strict avalanche criterion [22]. Thus the proposed method of concurrent error detection by parity

modification is effective in detecting technical as well as deliberately injected faults.

7. Acknowledgements

This research was performed while the first author visited University of Potsdam on an Alexander Von Humboldt Fellowship.

8. References

1. R. J. Anderson "Crypto in Europe – Markets, Law and Policy", *Cryptography: Policy and Algorithms*, Springer-Verlag LNCS 1029 pp. 75-89.
2. R. J. Anderson and M. Kuhn, "Low cost attack on tamper resistant devices", *Proceedings International Workshop on Security Protocols*, Springer-Verlag LNCS 1361, 1997.
3. G. Bertoni, L. Breveglieri, I. Koren, P. Maistri and V. Piuri, "A parity code based fault detection for an implementation of the advanced encryption standard," *Proceedings IEEE International Symposium on Defect and Fault Tolerance in VLSI*, pp. 51-59, Nov. 2002.
4. G. Bertoni, L. Breveglieri, I. Koren and V. Piuri, "On the propagation of faults and their detection in a hardware implementation of the advanced encryption standard," *Proc. of ASAP'02*, pp. 303-312, 2002.
5. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", *Journal of Cryptography*, Vol. 4, No. 1, pp. 3-72, 1991.
6. E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems", *Proc of Crypto*, 1997.
7. D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults", *Proceedings of Eurocrypt*, Springer-Verlag LNCS 1233, pp. 37-51, 1997.
8. H. Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, R. Zimmermann and W. Fichtner, "VINCI: Secure test of a VLSI high-speed encryption system", *Proc of International Test Conference*, pp. 782–790, Oct 1993.
9. A. S. Butter, C. Y. Kao and J. P. Kuruts, "DES encryption and decryption unit with error checking," US patent US5432848, Jul 1995.
10. J. Daemen and V. Rijmen, "AES proposal: Rijndael", http://www.esat.kuleuven.ac.be/~rijmen/rijndael/ rijndaeldocV2.zip
11. S. Fernandez-Gomez, J. J. Rodriguez-Andina and E. Mandado, "Concurrent Error Detection in Block Ciphers", *IEEE International Test Conference*, Oct 2000.
12. J. Stern and S. Vaudenay, "CS-Cipher," *Proceedings of Fast Software Encryption: 5th International Workshop*, Springer-Verlag LNCS 1372, pp. 189-205, Mar 1998.
13. 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 Transactions on CAD*, Dec 2002.
14. J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side-Channel Cryptanalysis of Product Ciphers," *Proceedings of ESORICS*, Springer-Verlag, pp. 97-110, Sep 1998.
15. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *Proceedings of CRYPTO-96*, Springer-Verlag, pp. 104-113, 1996.
16. P. Kocher, J. Jaffe and B. Jun, "Introduction to Differential Power Analysis and Related Attacks". http://www.cryptography.com/dpa/technical, 1998.
17. M. Matsui, "Linear Cryptanalysis Method for DES Cipher," *Proceedings of Eurocrypt*, Springer-Verlag, pp. 386-397, 1994.
18. R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, " The RC6 block cipher", ftp://ftp.rsasecurity.com/pub/rsalabs/aes/rc6v11.pdf
19. S. Vaudenay, "An experiment on DES Statistical Cryptanalysis", *3rd ACM Conference on Computer and Communications Security*, pp. 139-147, 1996.
20. S. Wolter, H. Matz, A. Schubert and R. Laur, "On the VLSI implementation of the International Data Encryption Algorithm IDEA", *IEEE Intl symposium on Circuits and Systems*, Vol.1, pp. 397-400, 1995.
21. R. Karri, M. Goessel and G. Kousnezow, "Method for error detection in kryptographic substitution permutation networks," Patent application pending.
22. A. F. Webster and S. E. Tavares. "On the design of S-boxes," *Proceedings of CRYPTO '85*, Springer Verlag Lecture Notes in Computer Science, LNCS 218, pp. 523—534, 1986.
23. C. Aumuller, P. Bier, P. Hofreiter, W. Fischer and J.-P. Seifert, "Fault attacks on RSA with CRT: concrete results and practical countermeasures," www.iacr.org/eprint/2002/072.pdf.
24. J. Bloemer and J.-P. Seifert, "Fault based cryptanalysis of the Advanced Encryption Standard," www.iacr.org/eprint/2002/075.pdf.