Parity-Based Concurrent Error Detection of Substitution-Permutation Network Block Ciphers

Ramesh Karri[1], Grigori Kuznetsov[2], and Michael Goessel[2]

[1]Department of Electrical and Computer Engineering
Polytechnic University, 6 Metrotech Center
Brooklyn, NY 11201

[2]Institute of Computer Science,
Fault Tolerant Computing Group
University of Potsdam
D-14439 Potsdam, Germany
ramesh@india.poly.edu,{grigoriy,mgoessel}@cs.uni-potsdam.de

Abstract. Deliberate injection of faults into cryptographic devices is an effective cryptanalysis technique against symmetric and asymmetric encryption algorithms. In this paper we will describe parity code based concurrent error detection (CED) approach against such attacks in substitution-permutation network (SPN) symmetric block ciphers [22]. The basic idea compares a carefully modified parity of the input plain text with that of the output cipher text resulting in a simple CED circuitry. An analysis of the SPN symmetric block ciphers reveals that on one hand, permutation of the round outputs does not alter the parity from its input to its output. On the other hand, exclusive-or with the round key and the non-linear substitution function (s-box) modify the parity from their inputs to their outputs. In order to change the parity of the inputs into the parity of outputs of an SPN encryption, we exclusive-or the parity of the SPN round function output with the parity of the round key. We also add to all s-boxes an additional 1-bit binary function that implements the combined parity of the inputs and outputs to the s-box for all its (input, output) pairs. These two modifications are used only by the CED circuitry and do not impact the SPN encryption or decryption. The proposed CED approach is demonstrated on a 16-input, 16-output SPN symmetric block cipher from [1].

1 Introduction

Until recently cryptanalysts analyzed cipher systems by using rigorous mathematics based techniques such as differential cryptanalysis [2] and linear cryptanalysis [3]. 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 by the operations, power dissipated by the operators, electromagnetic radiation emitted by the device and faulty computations resulting from deliberate injection of faults into the system. Traditional cryptanalysis techniques can be combined with such side-channel attacks to uncover break the secret key and/or break the implementation details of the cipher. Even a small amount of side-channel information is sufficient to break common ciphers [4]. For example, Differential Fault Analysis (DFA) that uses deliberate injection of faults requires between 50 to 200 cipher text blocks to recover a key of symmetric block cipher Data Encryption Standard (DES); the best traditional attack requires approximately 64 terabytes of plain text and cipher text encrypted under a single key.

1.1 Fault Based Attacks: Motivation

Fault based cryptanalysis (for example, DFA) is based on the observation that faults deliberately injected 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, DeMillo and Lipton [5] presented the first fault based side-channel attack against asymmetric public-key cryptography devices. More recently, Biham and Shamir [6] presented a fault-based cryptanalysis of symmetric block cipher Data Encryption Standard (DES). They presented a transient fault based Differential Fault Analysis (DFA) attack and a permanent fault based non-DFA attack to recover the round keys using a very small number of cipher texts. They then extended their fault model to show that DFA can uncover the structure of an unknown cryptosystem implemented in an EEPROM based smart card based on the observation that it is much easier to inject a 1→0 bit flip than to inject a 0→1 bit flip in an EEPROM. Using DES as the unknown cipher, they showed that (i) about 500 faulty cipher texts are sufficient to identify the bits of the right half, (ii) about 5000 faulty cipher texts are sufficient to identify the non-linear substitution operations (s-boxes) and their input and output bits, and (iii) about 10000 faulty cipher texts are sufficient to reconstruct the DES s-boxes.

Anderson and Kuhn described additional fault based side-channel attacks on software implementations of encryption algorithms [7]. 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. Another attack focused on the chip writing ability of the attacker. Assuming that the attacker is familiar with the implementation, he can extract keys from the card by overwriting specific memory locations.

1.2 Fault-Based Side Channels: The Fault Models

Boneh, Demillo and Lipton [5] use 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 the asymmetric property of EEPROMs: it is much easier to induce a 1→0 bit flip than to induce a 0→1 bit flip. Anderson and Kuhn used two different fault models for microcontroller based smartcards: in the first they assume that the instruction memory of smart cards can be randomly corrupted and in the second they assume 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 [8,9]. The proposed CED approach is applicable to the practical fault models described.

1.3 CED Architectures for Symmetric Block Ciphers: Background

Concurrent error detection (CED) followed by suppression of the corresponding faulty output can thwart fault injection attacks; 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 [10]. Spares based approaches are suitable for block ciphers that use arithmetic operators, such as IDEA and RC6 [11]. 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) [12] and DES that use random, non-arithmetic operations such as S-Boxes.

Time redundancy based CED approach involves encrypting (decrypting) the data a second time followed by the comparison of two results. Wolter et. al. [13] 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.

Karri et. al [14] developed a systematic CED approach for symmetric block ciphers at the register transfer level that exploits the inverse relationship between the 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 Advanced Encryption Standard, RC6 and Serpent. The main drawback of this approach is that it 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. [15] applied this inverse-relationship principle to round key generation of the AES encryption algorithm using additional hardware for inverse round key generation and comparison.

Another CED approach involves encoding the message before encryption and checking it for errors after decryption. Wolter et. al. [13] used residue codes for fault detection in adders, multipliers, and EXCLUSIVE-ORs. 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. In [16] the plaintext is encoded by setting several bits of the message to a particular fixed value, 0 or 1, and then encrypted. A mismatch between these fixed bits of deciphered text and the original plain text detects an error. The simple code (all zeroes or all ones) results in significantly less area overhead when compared to other encoding schemes. This scheme has a large fault detection latency detects since faults in the encryption hardware at the transmitter end by the decryption hardware at the receiver. Further, there is an associated performance penalty since it uses some of the bits in messages for error detection.

In [17] a CED technique that predicted the inverse of the parity of the outputs was proposed for the non-linear s-box and other functions used in DES. A similar technique that predicted the parity of the outputs for the non-linear s-box and linear mixing functions used in the AES was proposed in [18,19]. In these papers one additional parity bit per byte at the outputs of the s-boxes is added. To detect errors at the inputs to the s-boxes, the inputs of the s-boxes are also parity encoded. The size of the s-boxes is doubled by proposing a 512×9-bit implementation resulting in an area overhead of over 100% for s-box CED.

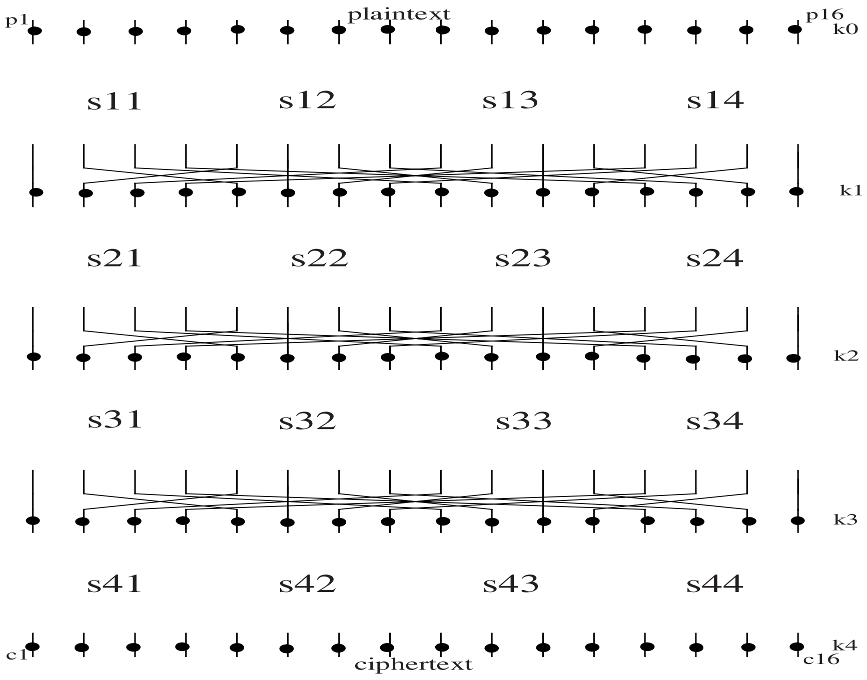

Fig. 1. Substitution Permutation Network (SPN) cryptosystem

2 Substitution-Permutation Network (SPN) Block Ciphers

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 to generate 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. All the rounds of encryption and decryption are identical in general, with each round using several operations and round key(s) to process the input data. Consider the well-known substitution-permutation network (SPN) cryptosystem shown in Figure 1. Such an SPN architecture consisting of a non-linear substitution layer (s-boxes) connected by output bit position permutations is an easy to understand yet realistic architecture [1]. The example SPN cryptosystem shown in Figure 1 operates on a 16-bit plaintext generating a 16-bit cipher text and four rounds. Each SPN encryption round is composed of a non-linear substitution operation (using four 4x4 s-boxes), a permutation and exclusive-or with a 16-bit round key. The sixteen 4x4 s-boxes in this example cryptosystem are different. To preserve symmetry between encryption and decryption, the first round operation is preceded by exclusive-or with the 16-bit key, key 0. Then the four 16-bit round keys (key 1, key 2, key 3, and key 4) are exclusive-ored following the permutation operation (In Figure 1 the dots on the s-box input lines represent exclusive or) in each round.

2.1 Parity-Based Concurrent Error Detection

Protection of crypto-devices entails protecting the encryption/decryption data paths as well as the key ram used to hold the round keys. Significant work has been done to protect the RAM using Parity code, Hamming code etc. In this paper we are interested in CED of the encryption data path and we do not address CED for key RAM.

The proposed CED design approach uses parity code. The specific CED implementation depends on the SPN implementation architecture. Consider the unfolded implementation architecture shown in Figure 1 (this is necessary because all 16 s-boxes are different). The parity of the inputs to the first round, P(x) is determined by a parity tree of the 16 inputs. The CED structure modifies this input parity according to the successive processing steps of the SPN round function such that the modified parity is equal to the parity of the outputs of the SPN circuitry of the first round. The CED architecture shown in Figure 2 repeatedly modifies the parity in the manner discussed in each of the four rounds and compares it with the parity of the cipher text.

The operations in an SPN round are: non-linear transformations by the sixteen 4x4 substitution boxes (s-box), bit-permutation and exclusive-or with the round key. Non-linear substitution boxes used in SPN-based and other encryption algorithms have been designed to satisfy properties such as maximum non-linear order, high nonlinearity, low differential uniformity and low bias [20,21]. Satisfaction of these properties has been shown to reflect the strength of the s-box against linear and differential cryptanalysis. These s-boxes do not maintain the parity from their inputs to their outputs.

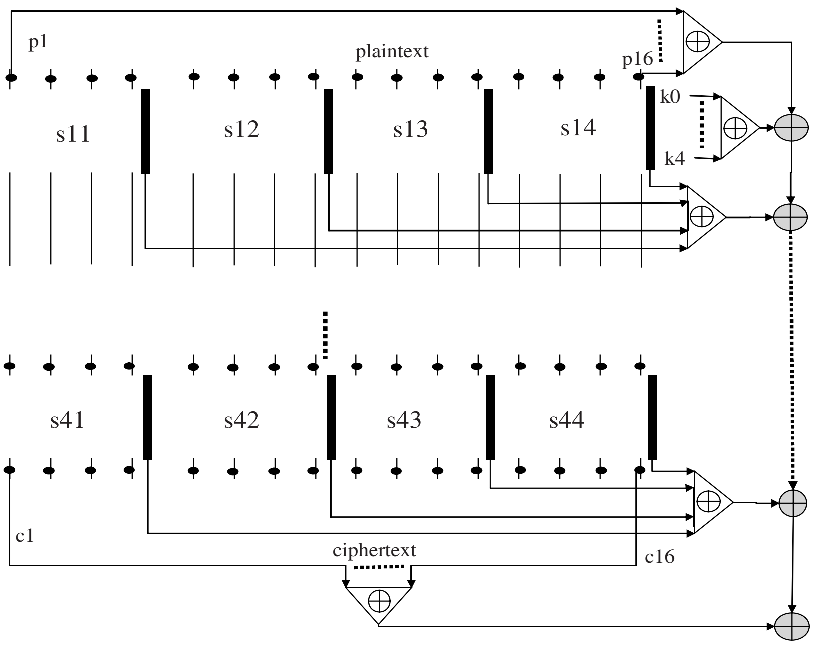

Fig. 2. CED architecture of the SPN block cipher

Table 1. 4x4 substitution box supplemented with m (i) =parity (i)⊕parity(s(i))

I	s(i)	parity(i)⊕parity(s(i))
0	E	1
1	4	0
2	D	0
3	1	1
4	2	0
5	F	0
6	B	1
7	8	0
8	3	1
9	A	0
A	6	0
B	C	1
C	5	0
D	9	1
E	0	1
F	7	1

We add to every four input, four output s-box an additional binary output for the purpose of modifying the input parity of the SPN circuitry into the output parity in the considered SPN round. Let s be an s-box with four inputs i1, i2, i3, i4 and four outputs s1(i1,i2,i3,i4), s2(i1,i2,i3,i4), s3(i1,i2,i3,i4), and s4(i1,i2,i3,i4). Then for every input i= i1,i2,i3,i4 of the s-box, the additional modifying output m(i) implements i1⊕ i2⊕ i3⊕ i4⊕ s1⊕ s2⊕ s3⊕ s4⊕ =parity(i)⊕parity(s(i)). Parity (i) and parity (s(i)) are the input parity and the output parity respectively of the considered s-box. The design of this modifying output m (i) for an example 4x4 s-box from [1] is shown now. The first two columns of Table 1 show the truth table of an s-box. In the first column the four-bit inputs i and in the second column the four-bit outputs s(i) of the s-boxes are given in hexadecimal representation. In the third column the binary value m(i) which implements m(i) = parity(i)⊕parity(s(i)) is given. Thus, for example in row 6 of Table 1 for the input i = i1,i2, i3,i4 = 0101 = 5 the functional output of the s-box is s(5) = s1(5),s2(5),s3(5), s4(5) = 1111 = F, and for the additional binary output m(5) of the s-box we have m(5) =0⊕1⊕0⊕1⊕1⊕1⊕1⊕1=0.

In the complete CED architecture shown in Figure 2, a thick box appended to the right hand side of an s-box shows this modifying output. This modifying output in each s-box is used only for CED and does not impact either the encryption or the decryption. Since we do not change the functionality of the s-boxes, the strength of the used cryptographic algorithm, based to a large extent on the concrete form of the s-boxes, is preserved.

Next, since permutations do not change the parity no modification circuitry is necessary. Finally, bit-wise modulo 2 addition of the 16-bit key 0 modifies the parity of the input plain text by parity of key 0 prior to the first SPN round. Bit-wise modulo 2 addition of the 16-bit key 1 modifies the parity of the input plain text by parity of key 1 in the first SPN round. Similarly, bit-wise modulo-2 addition of the 16-bit key 2 modifies the parity of the input plain text by parity of key 2 in the second SPN round and so on. The overall modification due to all the round keys can be pre-computed during round key generation as parity of key 0 ⊕ parity of key 1 ⊕ parity of key 2 ⊕ parity of key 3 ⊕ parity of key 4. This absorbs the associated time overhead into that of round key generation.

While this architecture might apply to most of the common examples, it doesn't necessarily apply to all cases. For example, not all architectures require an explicit decryption module; some block ciphers, DES being the most noteworthy example, looks practically identical regardless of the direction as long as the round keys are reversed. Also, while many block ciphers have some internal iterative round component, often in practice, they consist of other structures (such as pre and post-whitening steps). However, this general principle can be easily adapted to these situations. The proposed CED method is also applicable to other symmetric-key primitives such as message authentication codes and stream ciphers that have an SPN structure.

3 Fault Detection Capability

In section 2 we explained how the input parity of an SPN round is modified step-by-step according to the processing steps of that round in such a way that the modified parity of the inputs is equal to the parity of the outputs of that round if no error occurs.

In this section we show how an error due to a single stuck-at-fault in a processing step of the SPN Symmetric Block Cipher is detected by the proposed CED method. This is illustrated in Figure 3 for four successive processing steps. The inputs x are processed into the outputs y in step 1 and the parity P(x) of the inputs is modified into $P(x)\oplus P(x)\oplus P(y)=P(y)$. We assume now that a fault f occurs in the hardware implementing the processing step 2 with the result that the outputs of the second step are now z_f instead of the correct outputs z, with $z_f\neq z$. The parity P(y) is corrected in this second step into the correct value $P(y)\oplus P(y)\oplus P(z) = P(z)$. If the error due to the fault f is detectable by parity we have $P(z_f)\neq P(z)$. In step 3 the erroneous inputs z_f (instead of the correct inputs z) are correctly processed by the fault-free hardware of this third step into the outputs u_f and now the parity P(z) is modified into $P(z)\oplus P(z_f)\oplus P(u_f)$.

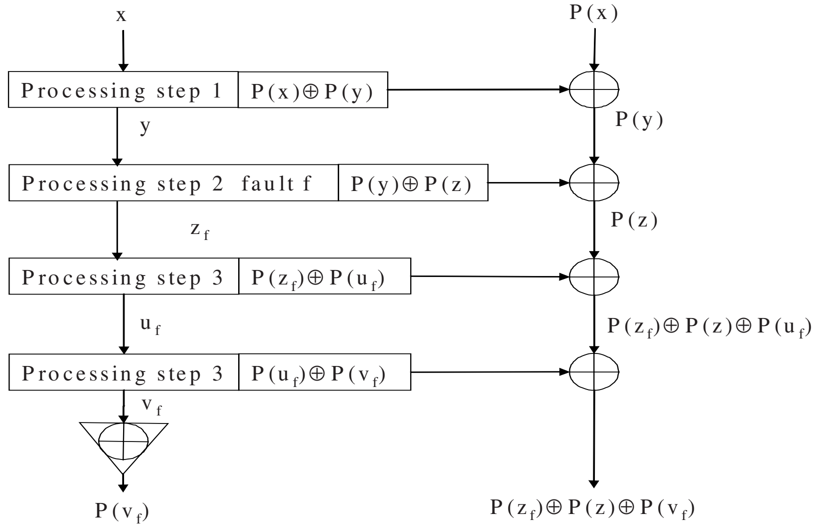

Fig. 3. Analysis of fault detection capability

Similarly in step 4 the inputs u_f are correctly processed into v_f and the parity $P(z)\oplus P(z_f)\oplus P(u_f)$ is now modified into $P(z)\oplus P(z_f)\oplus P(u_f)\oplus P(u_f)\oplus P(v_f)= P(z)\oplus P(z_f)\oplus P(v_f)$. Finally the modified parity $P(z)\oplus P(z_f)\oplus P(v_f)$ is compared with $P(v_f)$, the parity of the outputs v_f of step 4, and for $P(z) \neq P(z_f)$ the error due to the fault f will be detected. Thus, if, due to a fault f in step 2, a single bit (or an odd number of bits) is erroneous this error will be always detected by comparing the parity of the outputs of step 4 with the corresponding modified input parity.

Cryptographic algorithms are designed to satisfy the strict avalanche criterion [20,21]; even a single bit error at the inputs of an encryption step results in many different erroneous bits at the outputs of the following encryption steps. But, as explained, this property of encryption algorithms has no influence on the error detection capability of the proposed CED method.

The processing steps of the considered SPN Symmetric Block Cipher are compnent-wise exclusive-or with the round key, permutation and non-linear S-box transformation. For the operation exclusive-or with a round key every single (internal or external) stuck-at fault of an exclusive-or gate will result in a single bit error which is detected by parity. For the permutation operations a single stuck-at fault will result in a single bit error which is also detected by parity. For the 16 parity-appended S-boxes with four inputs and five outputs (four functional outputs and one parity modifying output) CED capability is described now. We designed these S-boxes using SISII logic synthesis tool from UC Berkeley. Then for all possible single stuck-at 0/1 faults the synthesized S-boxes were simulated for all possible input combinations. If all the outputs of the S-boxes are independently implemented (i.e. without sharing gates) every single stuck-at-fault results in a single bit error of the outputs of the corresponding S-box and will be obviously detected by parity. If all the five outputs of the S-boxes are jointly optimized then (in rare cases) even number of S-box output bits may be in error due to a single stuck-at fault. Then, as the experiments show, 96.3 % of the errors due to a single stuck-at fault are detected by parity. If the four functional outputs of the S-boxes are jointly optimized and if the parity-modifying bit is separately implemented 98.5% of the errors due to a single stuck-at fault are detected by parity. The area of an S-box with four inputs and four outputs without error detection in a two-level implementation is 56 units. With an additional fifth parity modifying output for CED the area, also in a two-level implementation, is 66 units. Thus, for a two-level implementation the area of the S-boxes increases by 18%. For a multi-level optimization the area of an S-box without CED is 41 units and with the additional parity modifying output it is 51 units. Thereby when the parity modifying output is separately implemented, and for a multi-level implementation the area of an S-box increases by 24.4%. Thus the overall hardware overhead is determined by an additional parity tree for computing the input parity; an 18% to 24.4% overhead for the implementation of the parity modification of the S-Boxes and some exclusive-or gates. As we have shown for a separate two-level implementation of the S-boxes with parity modification 100% error detection for all the errors due to single stuck-at faults is guaranteed by the proposed method.

3.1 Performance Penalty and Detection Latency

The parity of the input plaintext and output cipher text are computed for each plaintext and hence the associated delay should be carefully accounted for. Computing and checking of the parity can be combined with the round operations in several ways as shown in Table 2.

Table 2. Optimizing the latency of CED

Clock cycle	Approach A		Approach B		Approach C		Approach D	
	Round		Round		Round		Round	
1		P(PT)	1	P(PT)	1	P(PT)	1	P(PT)
2	1		2		2	P(1)	2	P(1)
3	2		3		3	P(2)	3	P(2)
4	3		4		4	P(3)	4	P(3)
5	4			P(CT)		P(CT)		
6		P(CT)						

In this table, we assume that encryption is implemented in four clock cycles with one round of encryption per clock cycle. In the straightforward approach A, the parity computation of the plaintext is followed by encryption (decryption) that in turn is followed by computing and checking the parity of the cipher text. This approach has a performance penalty of two clock cycles (one clock cycle for computing the parity of the input and one clock cycle for computing the parity of the cipher text + comparison with the input parity) and a fault detection latency of one complete encryption (decryption) i.e. four clock cycles.

In approach B, the parity of the plaintext is computed concurrently with the first round of encryption in clock cycle 1. This is then followed by the rest of the encryption (decryption) which in turn is followed by computation and checking of the parity of the cipher text. This approach reduces the performance penalty to one clock cycle without reducing the detection latency. In Approach C computation of the parity of the plaintext is performed concurrently with the first round of encryption in clock cycle 1. This is then followed by computation and checking of output of round one in parallel with the second round of encryption in clock cycle 2 and so on. This approach reduces the fault detection latency while maintaining the performance penalty of Approach B. Other approaches to absorbing performance penalty associated with CED are possible. Each approach has associated performance penalty, fault detection latency (the worst case duration between occurrence and detection of a fault) and fault coverage.

4 Conclusions

In this paper a new method for CED for SPN encryption Block Ciphers was proposed. More details on this general method can be found in [22]. Many of the well known symmetric block ciphers including AES [12] are SPN ciphers. According to the processing steps of the SPN network the parity of the inputs of an encryption round is modified into the parity of the outputs and compared with the actual parity of the outputs of this round. To reduce the necessary hardware overhead the parity tree for computing the parity of the inputs can be also be used to compute the parity of the outputs. If all functional outputs and the output for parity modification of the S-boxes are separately optimized a 100% error detection for all the errors due to single stuck-at faults was achieved. The additional area overhead is low. It consists of an additional parity tree for computing the parity of the inputs, a 18% to 24% increase of the area for the S-Boxes and a few exclusive-or gates only. The proposed concurrent error detection method allows detection of deliberately injected faults in addition to technical faults.

References

1. H. Heys, "A tutorial on linear and differential cryptanalysis," http://citeseer.nj.nec.com/443539.html.
2. E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Crytptosystems", *Journal of Cryptography*, Vol. 4, No. 1, pp. 3−72, 1991.
3. M. Matsui, "Linear Cryptanalysis Method for DES Cipher," *Proceedings of Advances in Cryptology-Eurocrypt*, Springer-Verlag, pp. 386−397, 1994.
4. J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side-Channel Cryptanalysis of Product Ciphers," *Proceedings of ESORICS*, Springer, pp. 97−110, Sep 1998.
5. D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults", *Proceedings of Eurocrypt*, Lecture Notes in Computer Science, Springer-Verlag, LNCS 1233, pp. 37−51, 1997.
6. E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems", *Proceedings of Crypto*, Aug 1997.
7. R. J. Anderson and M. Kuhn, "Low cost attack on tamper resistant devices", *Proceedings 5th International* Workshop on *Security Protocols*, Lecture Notes in Computer Sciences, Springer-Verlag, LNCS 1361, 1997.
8. 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.
9. J. Bloemer and J.-P. Seifert, "Fault based cryptanalysis of the Advanced Encryption Standard," www.iacr.org/eprint/2002/075.pdf.
10. 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
11. H. Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, R. Zimmermann and W. Fichtner, "VINCI: Secure test of a VLSI high-speed encryption system", *Proceedings of IEEE International Test Conference*, pp. 782–790, Oct 1993.
12. J. Daemen and V. Rijmen, "AES proposal: Rijndael", http://www.esat.kuleuven.ac.be/~rijmen/ rijndael/ rijndaeldocV2.zip
13. S. Wolter, H. Matz, A. Schubert and R. Laur, "On the VLSI implementation of the International Data Encryption Algorithm IDEA", *IEEE International symposium on Circuits and Systems*, Vol.1, pp. 397−400, 1995.
14. 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.
15. 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," *Proceedings of ASAP'02*, pp. 303−312, 2002.
16. S. Fernandez-Gomez, J. J. Rodriguez-Andina and E. Mandado, "Concurrent Error Detection in Block Ciphers", *IEEE International Test Conference*, Oct 2000.
17. A. S. Butter, C. Y. Kao and J. P. Kuruts, "DES encryption and decryption unit with error checking," US patent US5432848, Jul 1995.

18. 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.
19. G. Bertoni, L. Breveglieri, I. Koren, and V. Piuri, “Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard,” *IEEE Transactions on Computers*, vol. 52, No. 4, pp. 492−505, April 2003.
20. 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.
21. H. Heys and S. E. Tavares, “Avalanche characteristics of substitution permutation encryption networks,” *IEEE Transactions on Computers*, vol. 44, no. 9, pp. 1131−1139, Sep 1995.
22. R. Karri, M. Goessel, and G. Kousnezow, “Method for error detection in kryptographic substitution permutation networks,” patent application pending.