# Concurrent Error Detection of Fault-Based Side-Channel Cryptanalysis of 128-Bit Symmetric Block Ciphers

Ramesh Karri, Kaijie Wu, Piyush Mishra
ECE Department, Polytechnic University
5 Metrotech Center, Brooklyn, NY, 11201
1-718-260-4011
ramesh@india.poly.edu, kwu03,pmishr01@utopia.poly.edu

Yongkook Kim
IBM Corporation
Poughkeepsie, NY, 12601
1-845-435-7563
yongkook@us.ibm.com

**Abstract**: *Fault-based side channel cryptanalysis is very effective against symmetric and asymmetric encryption algorithms. Although straightforward hardware and time redundancy based concurrent error detection (CED) architectures can be used to thwart such attacks, they entail significant overhead (either area or performance). In this paper we investigate systematic approaches to low-cost, low-latency CED for symmetric encryption algorithms based on the inverse relationship that exists between encryption and decryption at algorithm level, round level and operation level and develop CED architectures that explore the trade-off between area overhead, performance penalty and error detection latency. The proposed techniques have been validated on FPGA implementations of AES finalist 128-bit symmetric encryption algorithms.*

## 1. Introduction

Hardware and software implementations of encryption algorithms leak information via side-channels such as measurement of time or power consumed by the operations used and deliberate introduction of faults. Rigorous mathematical analysis can be combined with such side-channel information to reveal the secret key and/or the implementation details of the encryption algorithms. These side-channel analysis attacks are much more powerful compared to mathematical analysis based attacks. Kelsey, Schneier, Wagner, and Hall showed that even a small amount of side-channel information is sufficient to break some of the common encryption algorithms [1].

Side-channel attacks can be defeated by carefully designing the software/hardware to either reduce the amount of side-channel information that leaks or make the leakage irrelevant. Denying an attacker the ability to monitor the internal states can defeat processor flag based side-channel attack on RC5 encryption algorithm [2] and Hamming weight based side-channel attack against Data Encryption Standard (DES) encryption algorithm [3].

Timing attacks use the timing characteristics of the implementation of operations in an encryption algorithm to break it [4, 5]. A simple counter measure for such an attack is to make the time for any operation independent of the input. Blinding is another counter measure that modifies the basic computation to produce correct result, but with the details of the modification invisible to the attacker. Differential Power Analysis (DPA) exploits the correlation between the power dissipation and bits of internal stage during encryption [6]. This attack does not require much knowledge of the implementation and yields both the key and enough information to derive a model of the device. Solutions to thwart DPA include masking the side-channel power information by performing random calculations, adding complementary circuits to mirror the real encryption calculations and varying the order of the operations in software.

Boneh, DeMillo and Lipton introduced fault-based side-channel attacks based on the observation that errors induced in the hardware devices leak information about the implemented encryption algorithm [7]. They showed that by exposing an encryption device to ionizing or microwave radiation, a fault could be induced at a random bit location in one of the registers at some random intermediate round in the cryptographic computation that can be used to easily factor the modulus of an RSA public key encryption algorithm. While the Number Field Sieve factoring developed by Lenstra and others have so far broken RSA implementations using maximum 155-digit (i.e., 431-bit) modulus [8], a fault-based attack can break RSA implementations using any length modulus.

Side channel attacks can be applied against a wide variety of applications, including pay-TV smart cards, prepayment meter tokens, intellectual proprietary rights violations and extraction of the secret information stored in a smart card [9]. Unfortunately, almost all existing counter measures against side-channel attacks suffer from a variety of drawbacks including large time and/or hardware overhead, severe performance degradation, the need to modify the implemented algorithm and ad-hoc nature.

## 2. Fault-based side-channel cryptanalysis

Soon after the first attack by Boneh et. al. a University of Singapore team proposed a fault-based attack against tamperproof RSA devices based on two fault models [10]. Biham and Shamir presented a fault-based side channel attack called Differential Fault Analysis (DFA) against

DES [11]. DFA can find the last DES round key using less than 200 cipher texts. Floyd et. al. developed a DFA attack on RC5 [12]. Biham and Shamir extended their fault model to show that DFA can uncover the structure of an unknown cryptosystem implemented in a smart card. Their fault model was based on the asymmetric properties of EEPROMs: inducing a $1 \rightarrow 0$ bit flip is much easier than inducing a $0 \rightarrow 1$ bit flip. Anderson and Kuhn described additional side-channel attacks against tamper resistant devices that can be launched with moderate difficulty [13].

Concurrent Error Detection (CED) followed by suppression of the corresponding output has been suggested as an approach to tolerate fault-based side-channel cryptanalysis – On detecting a faulty computation the key is protected by suppressing the cipher text. CED can be performed by straightforward duplication of encryption (decryption) hardware and comparison or by use of spares, though this scheme entails more than 100% hardware overhead. Based on this observation a hardware implementation of the 128-bit International Data Encryption Algorithm (IDEA) called VINCI implemented spares-based concurrent error detection [14]. However, it entailed significant hardware overhead and required very complex interconnections. Time redundancy based CED approach involves encrypting (decrypting) the data a second time using the same hardware, followed by a comparison of the two results. Wolter et. al. developed two CED schemes for IDEA based on information and time redundancy respectively [15]. These schemes are expensive and can tolerate only transient faults if the input traverses identical paths through the encryption (decryption) data path both the times.

Another CED approach involves encoding the message before encryption and checking it for errors after decryption [16]. This scheme results in significantly less area overhead as compared to other encoding schemes but has large fault detection latency and needs modification of the algorithm to improve its performance. Also, it cannot tolerate fault-based side-channel cryptanalysis since it assumes that the encrypted data is transmitted to the decryption module over a fault-proof channel and detects the fault, if any, only after decryption.

*In this paper, we investigate systematic approaches to low cost, low latency CED of symmetric encryption algorithms. These CED techniques exploit the inverse relationships that exist between encryption and decryption at various levels; any input data that is passed successively through encryption and decryption process/round/operation is recovered. These schemes offer better trade-off between area and time overhead without severely degrading the performance or modifying the encryption algorithm.*

## 3. Symmetric encryption algorithms

Symmetric encryption algorithms have an iterative looping structure as shown in Figure 1. After optional pre (post) processing, the input plain (cipher) text is subjected to one round of encryption (decryption) where a series of operations are performed on it and the round key(s) to generate the intermediate cipher text. The intermediate cipher text is then used as input to the next round. After a pre-determined number of rounds, the output is optionally post processed to generate the cipher (plain) text.

*Decryption is the inverse of encryption, with three levels of inverse relationships. First inverse relationship is at the algorithm level, where the order of rounds in decryption is the reverse of that in encryption. The second inverse relationship is at the round level wherein the sequence of operations in a round of decryption is the reverse of the sequence of operations in a round of encryption. For example, if a round of encryption starts with operation 1 and goes on to operation m, the corresponding round of decryption starts with operation m and goes on to operation 1. The final inverse relationship is at the operation level with decryption using the corresponding inverse operations of encryption.*

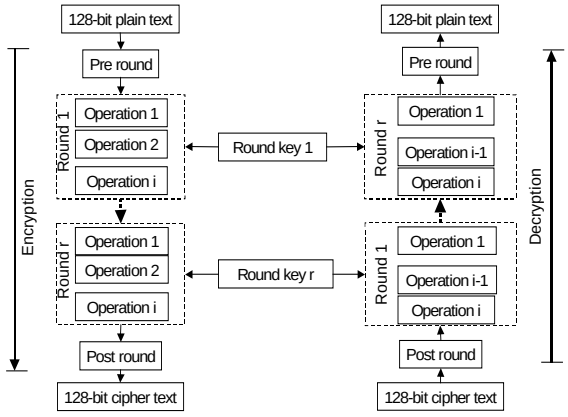


**Figure 1: 128-bit symmetric encryption algorithm**

We next discuss Advanced Encryption Standard (AES) finalist symmetric encryption algorithms within this framework. All AES algorithms support multiple key lengths (128, 192 and 256 bit key).

Serpent is a 128-bit encryption algorithm that consists of 32 rounds and a pre and post-processing step [19]. Each round uses 1 round key (except for the last round, which uses 2 round keys). Exclusive-or of 128-bit round key with 128-bit round input, non-linear substitution, and linear transformation that performs bit-wise exclusive-or on selected input bits are the operations used in a round of Serpent encryption.

128-bit Twofish encryption comprises of 16 rounds with each round using 2 round keys [20]. Four keys each are used during pre and post processing steps. The operations in an encryption round are key-dependent non-linear substitution, GF ($2^8$) matrix multiplication, a mixing operation based on pseudo-Hadamard transformation, key addition and bit-wise rotation.

RC6 encryption algorithms support multiple data block sizes. 128-bit RC6 encryption supports 20 rounds with each round using 2 round keys [17]. 4 additional keys are used during pre and post rounds. Each round of RC6 encryption uses two multiplications, two additions, two exclusive-or operations, two fixed rotations and two variable rotations.

**Table 1: Operations used by 128-bit symmetric encryption algorithms**

| | | | | | | |
|---|---|---|---|---|---|---|
| RC6 | $\times(\text{mod } 2^{32})$ | 5-bit rot. | $\oplus$ | Variable rot. | $+(\text{mod } 2^{32})$ (key) | |
| Rijndael | S-box | Fixed rot. | $\times(\text{GF } (2^8))$ | $\oplus$ (key) | | |
| Serpent | $\oplus$ (key) | S-box | $\oplus$(Lin. Transf.) | | | |
| Twofish | S-box | $\times(\text{GF } (2^8))$ | $+(\text{mod } 2^{32})$ | $+(\text{mod } 2^{32})$ (key) | $\oplus$ | 1-bit rot. |

Rijndael encryption algorithms also support multiple block sizes. A 128-bit block, 128-bit key Rijndael encryption supports 10 rounds, each round using 1 round key [18]. An additional key is used in pre-processing. Rijndael operates on a two-dimensional table of plain text bytes called the state. Operations used in a round of Rijndael are: a non-linear byte substitution operation (byte sub), a cyclic left shift of the rows in the state (shift row), GF ($2^8$) multiplication with a constant of every column of the state (mix column) and exclusive-or of round key with the state (key-xor).

Table 1 summarizes various operations used by these symmetric encryption algorithms in their encryption rounds. We have tried to present the operations in the order they are used within a round. In some encryption algorithms the order of the operations and the order of the keys in decryption is the exact inverse of that in encryption. Rijndael and Serpent decryption use operations that are inverse of the operations used by their respective encryption. Detailed description of each of these algorithms can be found in [17, 18, 19, 20].

## 4. CED of symmetric encryption algorithms

A typical encryption device consists of an encryption module, a decryption module, a key RAM, an input port and an output port. Since a symmetric encryption algorithm uses the same set of round keys for both encryption and decryption, they can be generated a priori, stored in the key RAM and retrieved in any order depending upon whether encryption or decryption is in progress. We assume that either encryption or decryption is performed at a time, which implies that the other module is idle and can be used for CED. Our proposed scheme combines this fact with the inverse properties of the symmetric encryption algorithm.

### 4.1 Algorithm level CED

Algorithm level CED approach shown in Figure 2 exploits the inverse relationship at the algorithm level. Plain text is processed through the encryption module, which is then disabled (or processes next block of data) and the decryption module is enabled to decrypt the cipher text. The output of decryption is compared with the input plain text. An error signal is set and the faulty cipher text is suppressed when there is a mismatch. Algorithm level CED during decryption is similar. The area overhead includes an additional register to store the original input, a comparator module, and a few multiplexers at the input of the encryption, decryption and comparator modules. Since the round keys are stored in a key RAM, next block of plain text (cipher text) cannot be processed until both encryption and decryption of current plain text (cipher text) is finished (since otherwise different round keys need to be accessed simultaneously by the encryption and decryption modules).

Hence, the time overhead of encryption for one block of data with algorithm level CED is the time it takes for encryption (without CED) of one block of data. The time to encrypt *N* blocks of input data using algorithm level CED is *2N* $\times$ the time for basic encryption of one block of data. Although this is identical to the time overhead of encryption (decryption) with time redundancy based CED, it can detect permanent faults as well. If the keys are stored in a register file or are stored in two different key RAMs - encryption key RAM and decryption key RAM, encryption of the current block of data can be carried out concurrently with the decryption (for CED) of the previous block of data. This reduces the total time for encrypting *N* blocks of data to *(N+1)* $\times$ the time for basic encryption of one block of data.

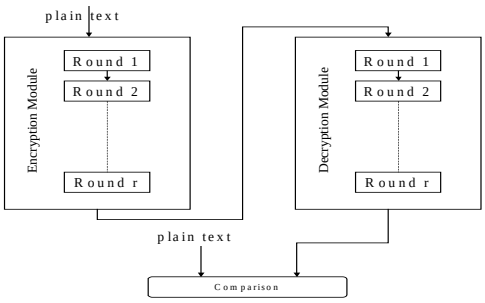


**Figure 2: Encryption with algorithm level CED**

Let us now consider fault detection latency, *the duration between the occurrence and detection of a fault*, for this CED scheme. For an encryption algorithm that has *r* rounds and *n* clock cycle(s) per round, the fault detection latency in the worst case is $2 \times n \times r$ .

### 4.2 Round Level CED

A closer look at the symmetric encryption algorithms reveals that the inverse relationship between encryption and decryption exists at the round level as well. *Passing the input data successively through one encryption round and the corresponding decryption round yields the original data.* For almost all the symmetric encryption algorithms, the first round of encryption corresponds to the last round of decryption; the second round of encryption corresponds to the last but one round of decryption and so on. Based on this observation, the computations can be checked at the round level. At the beginning of each encryption round corresponding round key and the input data is stored in registers before feeding to the round module. After one round of encryption is over, the output is fed to the corresponding round of decryption. The output of decryption round is calculated using the stored key and then compared with the input data that was previously saved. If there is a mismatch, the encryption process is halted and an error is

signaled. Encryption with round level CED is shown in Figure 3.

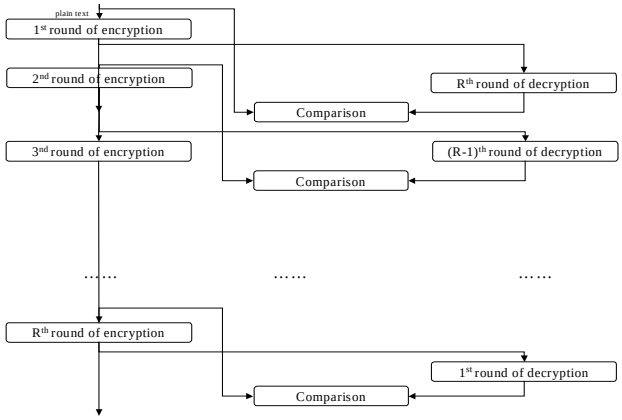


**Figure 3: Encryption with round level CED**

The performance penalty for encrypting one block of data with round level CED is now only a round, that is, *n* clock cycles. This is because the current round of decryption (for CED) can start concurrently with the next round of encryption. Further, the fault detection latency in the worst case is twice the time required for one round. Since each round takes *n* clock cycles, this equals $2 \times n$. The area overhead of round level CED is due to the additional registers, comparators, multiplexers and complex control. Table 2 shows the rounds of encryption and the corresponding rounds of decryption for each of the AES symmetric encryption algorithm. For example, in Serpent (32-round implementation), "round i" of encryption corresponds to "round (33-i)" of decryption and vice versa.

**Table 2: Encryption and corresponding decryption rounds of AES encryption algorithms satisfying the inverse relationship**

| RC6 | | Serpent | | Twofish | | Rijndael | |
|---|---|---|---|---|---|---|---|
| Enc | Dec | Enc | Dec | Enc | Dec | Enc | Dec |
| Pre Whiten | ost lacken | | | | | | |
| Whiten rnd i | lacken nd 21– i | rnd i | rnd 33 – i | rnd i | rnd 17-i | rnd i | rnd 11 –i |
| Post Whiten | re lacken | | | | | | |

### 4.3 Operation level CED

Depending on the encryption algorithm and its hardware implementation each encryption (decryption) round can be partitioned into operations (with each operation consuming one or more clock cycles) such that the operations of encryption and corresponding operations of decryption satisfy the inverse relationship. Consequently, *passing the input data through an operation in the encryption round and the corresponding inverse-operation in decryption round yields the original input data*. This is shown in Figure 4. The dotted box on the left shows the $r^{th}$ encryption round while the dotted box on the right shows the $(R\text{-}r+1)^{th}$ decryption round, where 'R' is the total number of rounds in encryption/decryption.

Further, Figure 4 shows that the first operation of the encryption round corresponds to the $m^{th}$ (i.e. last) operation of the decryption round. Such an operation level CED further improves the fault detection latency. In addition, it localizes the fault to the hardware implementing the operation.

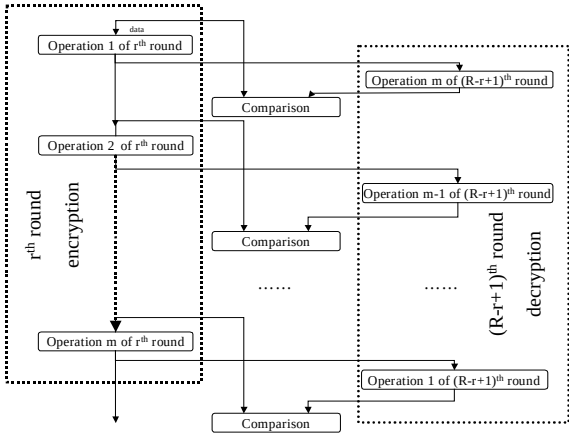


**Figure 4: Encryption with operation level CED**

On the other hand, complexity of the design increases and there is an additional performance penalty because of the large delay due to additional multiplexers.

**Table 3: Encryption and corresponding decryption operations of AES encryption algorithms satisfying the inverse relationship**

| RC6 | | Serpent | | Rijndael | |
|---|---|---|---|---|---|
| Enc | Dec | Enc | Dec | Enc | Dec |
| Op. 1 | Op. 1 | Xor | Xor | S-box | $S\text{-box}^{-1}$ |
| Op. 2 | Op. 2 | S-box | $S\text{-box}^{-1}$ | Shift-row | Shift- $row^{-1}$ |
| | | LT | $LT^{-1}$ | Mix column | Mix- $column^{-1}$ |
| | | | | Key-xor | Key-xor |

The operation level correspondence between encryption and decryption for the AES symmetric encryption algorithms is summarized in Table 3. For Serpent and Rijndael, operations shown in the column "Enc" are inverses of the operations shown in the column "Dec". For RC6 first operation of "Enc" is identical to (and not the inverse of) the first operation of "Dec". In this case, fault detection is achieved by performing computation on the encryption and decryption hardware and by comparing the two results. However, the second operation of encryption and decryption are inverses of one another. Twofish decryption uses the fact that the one-half of the output is same as the one-half of the input to any encryption round and the relation: a xor b xor b = a. Since there are no inverse operations involved, the only way of applying operation level CED will be to duplicate hardware, which is costly and hence not implemented.

Table 4 summarizes the performance and fault detection latency in terms of clock cycles for different CED architectures of AES encryption algorithms. The duration of a clock cycle differs from one encryption algorithm to the

next. Further, for a given encryption algorithm it also differs from one architecture to another. For example, in our implementation, one round of RC6 consumes two clock cycles. This translates into 42 clock cycles for a basic implementation of 20-round RC6 encryption (20 × 2 + 1 cycle for pre-processing +1 cycle for post-processing). Number of clock cycles for other algorithms and their respective CED architectures shown in table 4 has been computed similarly.

**Table 4: Comparison of CED architectures of 128-bit symmetric encryption algorithms**

| Algo. | No CED | Algorithm-level CED | | Round-level CED | | Operation-level CED | |
|---|---|---|---|---|---|---|---|
| | # of cycl. | # of cycl. | Det. Lat. | # of cycl. | Det. Lat. | # of cycl. | Det. Lat. |
| RC6 | 42 | 84 | 84 | 44 | 4 | 43 | 2 |
| Serpent | 64 | 128 | 128 | 66 | 4 | 65 | 2 |
| Twofish | 34 | 68 | 68 | 36 | 4 | 35 | 2 |
| Rijndael | 44 | 88 | 88 | 48 | 8 | 45 | 2 |

## 5. FPGA Implementation based validation

To validate the proposed CED techniques, we implemented the 128-bit symmetric encryption algorithms with different CED mechanisms in Xilinx Virtex device XCV1000BG560-6 FPGA (the largest FPGA currently available). All CED architectures were modeled in VHDL and functionally verified using Modeltech's Modelsim VHDL simulator. Synplify was then used for synthesis and Xilinx Foundation PAR tool was used for place and route. The results are presented in Table 5. Area is computed as the number of Virtex slices used. One Virtex Slice contains two look-up tables and one look-up table can implement four input-one output logic functions.

The throughput of an encryption algorithm represents the total number of data bits encrypted per second and is calculated as: (no. of bits encrypted) / (no. of clock cycles for encryption × clock duration). The performance degradation is obtained as: 1– (throughput of CED architecture/ throughput of basic architecture). From Table 5 it can be seen that the clock frequency, throughput and fault detection latency decrease while the area overhead increases with the granularity of CED. *Decrease in fault detection latency/increase in area* and *decrease in fault detection latency/decrease in throughput* ratios are more significant between algorithm level CED and round level CED than it is between round level CED and operation level CED.

## 6. CED Case Study: 128-bit Rijndael encryption

In our implementation, one round of Rijndael encryption (decryption) consumes four clock cycles. Therefore, encrypting (decrypting) one 128-bit block of plain (cipher) text consumes 11 x 4 = 44 clock cycles. The round keys are stored in a register file for use during both encryption and decryption. The area overhead for algorithm level CED is one register to store the plain or the cipher text, one comparator and couple of multiplexers. Fault detection latency using this approach is 44×2 = 88 clock cycles. Round level CED has a fault detection latency of 8 clock cycles, 4 cycles each for a round of encryption and decryption (Figure 5). Further, since the comparator is in the critical path module, the clock duration increased from 21.3ns to 27.74ns. Operation level CED is implemented for s-box, mix column and key XOR operations. Figure 6 shows the CED circuit for s-box and inverse s-box pair. Identical architectures will be used for the other two operations. The fault detection latency for the operation level CED will now be reduced to two clock cycles. Another benefit of operation level CED is that we can identify the faulty operation module pair.

**Table 5: Summary of FPGA implementation of CED architectures of 128-bit symmetric encryption algorithms**

| | | No CED | Algorithm Level CED | Round Level CED | Operation Level CED |
|---|---|---|---|---|---|
| Area (# of slices) | Rijndael | 3973 | 4806 | 4724 | 5486 |
| | RC6 | 2397 | 3028 | 3153 | 3337 |
| | Twofish | 3262 | 3474 | 3467 | N/A |
| | Serpent | 8073 | 9376 | 9659 | 9974 |
| Max freq (MHz). | Rijndael | 46.93 | 36.44 | 37.60 | 36.69 |
| | RC6 | 23.99 | 21.76 | 20.740 | 16.87 |
| | Twofish | 20.16 | 18.98 | 19.072 | N/A |
| | Serpent | 28.638 | 30.369 | 26.267 | 26.759 |
| Throughput (Mbps) | Rijndael | 136.53 | 53.04 | 100.27 | 104.36 |
| | RC6 | 73.11 | 33.16 | 60.33 | 50.22 |
| | Twofish | 75.90 | 35.73 | 67.80 | N/A |
| | Serpent | 57.28 | 30.37 | 50.95 | 52.69 |
| Area Overhead (%) | Rijndael | - | 20.97 | 18.90 | 38.08 |
| | RC6 | - | 26.3 | 31.5 | 39.20 |
| | Twofish | - | 6.49 | 6.28 | N/A |
| | Serpent | | 16.14 | 19.15 | 23.55 |
| Perf. Degrad.(%) | Rijndael | - | 61.15 | 26.55 | 23.56 |
| | RC6 | - | 54.64 | 17.48 | 31.31 |
| | Twofish | - | 52.92 | 10.67 | N/A |
| | Serpent | | 46.98 | 11.05 | 8.01 |

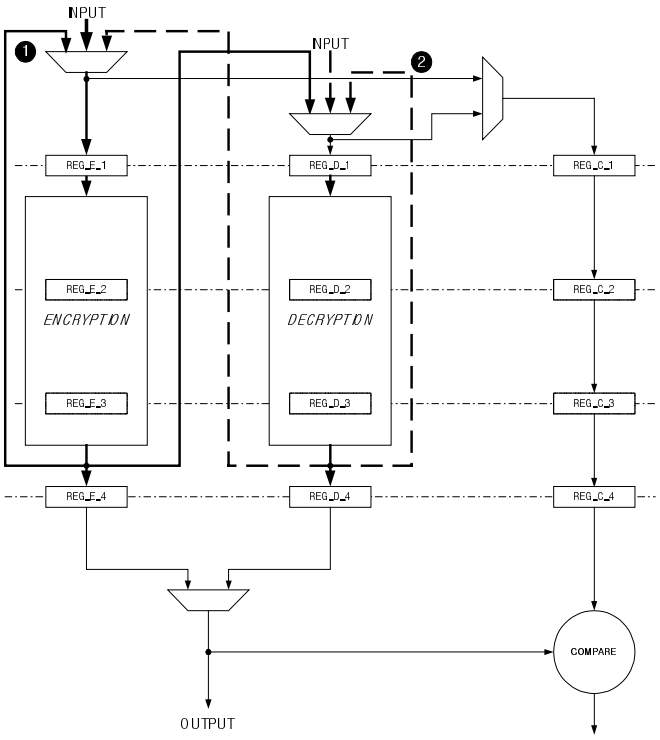


**Figure 5: Rijndael round level CED architecture**

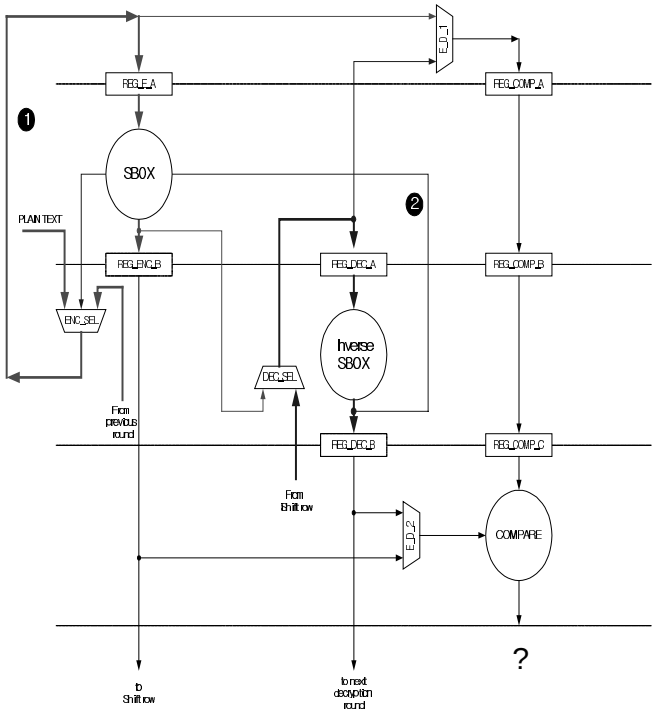


**Figure 6: Rijndael operation level CED of s-box/s-box-inverse**

## 7. Conclusions

We have presented algorithm level, round level and operation level CED architectures for symmetric encryption algorithms. Based on the FPGA implementations, we conclude that round level CED architectures better optimize the area overhead, performance penalty and fault detection latencies. However, operation level CED should be chosen when fault localization is important. From among all the AES symmetric encryption algorithms, Rijndael and Serpent are good candidates for implementing algorithm level, round level and operation level CED. Proposed scheme introduces moderate area overhead and interconnect complexity to achieve permanent as well as transient fault tolerance. This approach assumes that the key RAM, comparator or both encryption and decryption modules simultaneously are not under attack or faulty.

## 8. References

1. J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side Channel Cryptanalysis of Product Ciphers", Proceedings of ESORICS '98, Springer-Verlag, September 1998, pp. 97-110.

2. C. Harpes and J. Massey, “Partitioning Cryptanalysis”, Fast Software Encryption, 4th International Workshop Proceedings, Springer-Verlag, 1997, pp. 13-27.

3. S. Vaudenay, “An experiment on DES Statistical Cryptanalysis”, 3rd ACM Conference on Computer and Communications Security, ACM Press, 1996, pp. 139-147.

4. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", Proceedings of Advances in Cryptology-CRYPTO `96, Springer-Verlag, 1996, pp. 104-113.

5. J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestré, J. J. Quisquater and J. L. Willems, "A Practical Implementation of the Timing Attack", Proceedings of CARDIS, September 1998.

6. P. Kocher, J. Jaffe, B. Jun, "Introduction to Differential Power Analysis and Related Attacks", 1998. http://www.cryptography.com/dpa/technical

7. D. Boneh, R. DeMillo, and R. Lipton, “On the importance of checking cryptographic protocols for faults”, Proceedings of Eurocrypt, Vol. 1233, Springer-Verlag, 1997, pp. 37-51.

8. A. K. Lenstra, E. R. Verheul, “Selecting Cryptographic Key Sizes”, Public Key Cryptography Conference. http://www.cryptosavvy.com/cryptosizes.pdf

9. R. J. Anderson “Crypto in Europe – Markets, Law and Policy”, Cryptography: Policy and Algorithms, Springer LNCS v 1029 pp. 75-89. ftp://ftp.cl.cam.ac.uk/users/rja14/queensland.ps.Z

10. F. Bao, R. Deng, Y. Han, A. Jeng, T. Nagir, D. Narasimhalu, “A New Attack to RSA on Tamperproof Devices”, 5th International Workshop on Security Protocols, 1997, pp 125-136.

11. E. Biham, A. Shamir, “Differential Fault Analysis of Secret Key Cryptosystems”, Proceedings of Crypto'97, 1997.

12. J. J. Floyd, K.E. Fu, P. Sun, MIT, “6.857 Computer & Network Security Final Project: Differential Fault Analysis”, December 1996. http://web.mit.edu/jered/www/publications/rc5-dfa-paper.ps

13. R. Anderson, M. Kuhn, “Low cost attack on tamper resistant devices”, 5th International Workshop on Security Protocols, 1997. http://www.cl.cam.ac.uk/ftp/users/rja14/tamper2.ps.gz

14. H. Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, R. ZImmermann, W. Fichtner, “VINCI: Secure test of a VLSI high-speed encryption system”, Proceedings of IEEE International Test Conference, 1993, pp. 782 –790.

15. 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, volume 1, 1995, pp. 397-400.

16. S. Fernandez-Gomez, J. J. Rodriguez-Andina, E. Mandado, “Concurrent Error Detection in Block Ciphers”, IEEE International Test Conference, 2000.

17. R. L. Rivest, M. J. B. Robshaw, R. Sidney, Y. L. Yin, “The RC6™ block cipher”, ftp://ftp.rsasecurity.com/pub/rsalabs/aes/rc6v11.pdf

18. J. Daemen, V. Rijmen, “AES proposal: Rijndael”, http://www.esat.kuleuven.ac.be/~rijmen/ rijndael/ rijndaeldocV2.zip

19. R. Anderson, E. Biham, L. Knudsen, “Serpent: A proposal for the Advanced Encryption Standard”, http://www.cl.cam.ac.uk/ftp/users/rja14/serpent.tar.gz

20. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, “Twofish: A 128-bit cipher”, http://www.counterpane.com/twofish.pdf