Polytechnic
UNIVERSITY

Concurrent Error Detection Schemes for Fault Based Side-Channel Cryptanalysis of Symmetric Block Ciphers

Ramesh Karri, Piyush Mishra, Kaijie Wu, Polytechnic University
Yongkook Kim, IBM Corporation

November 2002

WICAT TR 02-006

Concurrent Error Detection Schemes for Fault Based Side-Channel Cryptanalysis of 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 overheads (either area or performance). In this paper we investigate systematic approaches to low cost, low latency CED techniques for symmetric encryption algorithms based on inverse relationships that exist between encryption and decryption at algorithm level, round level and operation level and develop CED architectures that explore tradeoffs among area overhead, performance penalty and fault detection latency. The proposed techniques have been validated on FPGA implementations of Advanced Encryption Standard (AES) finalist 128-bit symmetric encryption algorithms.

* This work is supported by NSF CAREER award CCR 996139

1. Introduction

Until recently crypto-analysts analyzed cipher systems by modeling them as ideal mathematical objects and applying conventional techniques such as differential cryptanalysis [1] and linear cryptanalysis [2]. Although these techniques are very useful in exploring weaknesses in algorithms, they do not exploit weaknesses in their software/hardware implementations. Implementation of (crypto) algorithms, either in hardware or software, leaks information via side-channels such as time consumed or power dissipated by the operations used, electromagnetic radiation emitted by the device and faulty computations due to the deliberately induced faults. Mathematical analysis techniques can be combined with such side-channel information to reveal the secret key and/or the implementation details of the cipher. These rigorous side-channel analysis attacks are much more powerful compared to the pure mathematical techniques. Kelsey, Schneier, Wagner, and Hall showed that even a small amount of side-channel information is sufficient to break some of the common ciphers [3]. For example, Differential Fault Analysis (DFA) side-channel attack requires between 50 to 200 cipher text blocks (no plain text) to recover a key of symmetric block cipher Data Encryption Standard (DES), while 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. Some of the examples include intellectual proprietary rights violations (for example, reverse engineering the pay-TV smart cards, pre-payment meter tokens, remote locking devices for cars or SIM cards for GSM mobile phones [13]), extraction of the secret information (such as the pin number, health records, and bank history) stored in a small smart cards etc. Side-channel attacks can be thwarted by carefully designing the software/hardware to either reduce the amount of side-channel information that leaks, such as using appropriate shielding to reduce electromagnetic radiation, or by making the leakage irrelevant, such as by using techniques that perform redundant, random computations to defeat timing and power attacks.

Eliminating side-channel information or preventing it from being used to attack a secure system is an active area of research. Several side-channel attacks and ad hoc counter measures have been proposed. Denying an attacker the ability to monitor the internal states can defeat processor flag based side-channel attack against RC5 encryption algorithm [4] and Hamming weight based side-channel attack against DES encryption algorithm [5]. Timing attacks exploit timing characteristics of the implementation of operations used in a cipher to break it [6, 7]. A simple counter measure for such

attacks is to make the time for any operation independent of the input [6]. Another counter measure involves Blinding. Blinding modifies the basic computation in such a way that it produces correct results but with the details of the modification invisible to the attacker [6]. Differential Power Analysis (DPA) attack divides 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 [8]. 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 implementations. Electromagnetic radiation based attacks utilize the fact that the application of square wave signals and high switching frequencies in digital equipment leads to radiation of electromagnetic fields containing frequency components up into the Ultra-high frequency (UHF) region, which can be used to reconstruct the original signal or data being processed by the device [9]. Counter measures against such radiation attacks include modifying devices and instruments by changing procedures and circuitry, reducing radiation level, heterodyning by noise or signals from external sources, shielding, interlocking, and filtering [10].

A relatively new and more serious threat is fault based side-channel cryptanalysis that is based on the observation that errors induced in the crypto-devices leak information about the implemented crypto-algorithms. These attacks are very practical since it is reasonable to assume that certain levels of radiation or heat, incorrect voltage, or atypical clock rate could be imposed on tamperproof devices. Such a physical stress can cause the device to malfunction and leak side-channel information. Several off-line and on-line counter measures have been proposed, though on-line schemes are more common for systems with high availability requirements.

Unfortunately, existing counter measures against side-channel attacks suffer from a variety of drawbacks, including large time and/or hardware overhead, severe performance degradation, and need to alter the algorithm implementation (which consumes significant effort). In addition, they are ad hoc techniques. In this paper we will investigate systematic approaches to low-cost, low-latency counter measures against the fault based side-channel cryptanalysis of symmetric encryption algorithms and develop architectures to explore the trade-offs among area overhead, performance penalty and fault detection latency.

2. Fault based side-channel cryptanalysis

Boneh, DeMillo and Lipton presented the first fault based side-channel attack against devices that use public-key cryptography [11]. For example, to sign a message X using RSA the encryption device computes $X^s \bmod N$, where s is a secret exponent. The security of the system relies on the fact that factoring the modulus N, a product of two primes p and q, is hard. When the factors of N are known the system can be easily broken. An efficient implementation of modular exponentiation $E = X^s \bmod N$ first computes $E_1 = X^s \bmod p$ and $E_2 = X^s \bmod q$ and then obtains $E = aE_1 + bE_2$ (where a and b are calculated using the Chinese Remainder Theorem). By exposing such a RSA crypto 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 of the cryptographic computation. Without loss of generality let E_1^f be the faulty result. Factor q can then be calculated as gcd (a $(E_1-E_1^f)$, N). This is a more powerful approach than other cryptanalysis techniques, such as the one using factoring. Number Field Sieve (NFS) factoring technique developed by Lenstra and others have so far broken RSA implementations that use 155-digit (i.e., 431-bit) modulus [12], while a fault based attack can be successfully applied to the modulus of any length. A University of Singapore team presented another fault based side-channel attacks against tamperproof RSA devices [14]. Their fault model assumes a bit flip in the secret exponent or in the cipher text and compares the faulty plain text with the correct plaintext to localize the faulty bit and identify its value.

More recently, Biham and Shamir presented a fault based side-channel cryptanalysis of DES symmetric block cipher [15]. DES divides 64-bit data block into 32-bit left and right blocks and applies sixteen rounds of encryption, with each round using a distinct round key. These round keys are derived from the secret key using the key expansion operation. 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. DFA attack assumes that one bit of data in one of the 16 rounds is flipped with a uniform probability and tries to identify that round to recover the last round key. Non-DFA attack inflicts a permanent fault, for example sticking the LSB of the register containing left-half of data to 0, and analyzes the cipher to recover the last round key. After which both the attacks can proceed in two ways: since the last round key contains 48 of the 56 bits of the user key, the attacker may either try all the possible remaining 2^8 combinations or peel off the last round of encryption to analyze the preceding rounds. Floyd, Fu and Sun presented a similar DFA attack on RC5

(a precursor to RC6) by introducing the register faults and then comparing the faulty result with the correct one to obtain the round keys [16]. They found the round key of last round first, then the round key of the round before last, and so on.

Biham and Shamir extended their fault model to show that DFA can uncover the structure of an unknown crypto-system implemented in the smart card. This attack was based on the following asymmetric properties of EEPROMs: it is much easier to induce a $1 \rightarrow 0$ bit flip than to induce a $0 \rightarrow 1$ bit flip. This attack used DES as the unknown cipher and required only about 500 faulty cipher texts to identify the bits of the right half, up to 5000 faulty cipher texts to identify the S-boxes (non-linear substitution operation) and their input and output bits, and about 10000 faulty cipher texts to reconstruct the DES S-boxes.

Anderson and Kuhn described additional fault based side-channel attacks in their paper on tamper resistance [17]. In one of the attacks, they assumed that the instruction memory of the smart cards could be corrupted. If in a process loop variable controlling the number of rounds is changed to the value 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.

Very little of the published literature discusses systematic techniques for protection against fault based side-channel cryptanalysis. Concurrent error detection (CED) followed by suppression of the corresponding faulty output has been suggested as an approach to tolerate fault based side-channel cryptanalysis; on detecting a faulty computation key is protected by suppressing the corresponding faulty cipher text. CED can be performed by straightforward duplication and comparison of encryption and decryption hardware yielding more than 100% hardware overhead. In a custom CED approach a spare module of each type is used to detect faults in all hardware modules of that type. VINCI, a hardware implementation of the 128-bit symmetric block cipher IDEA [18], used one spare module of each type of hardware unit (for example, a spare multiplier, adder etc.) for CED. Spares based approach is suitable only 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) and DES that use 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, Matz, Schubert and Laur developed a CED

technique for symmetric block cipher IDEA [19] wherein the test data was encrypted and then decrypted. This approach entails more than 100% time overhead. Further, it can tolerate only transient faults if the input traverses identical paths through the encryption and decryption data paths both the times.

Another CED approach involves encoding the message before encryption and checking it for errors after decryption. Wolter, Matz, Schubert and Laur developed a CED scheme for symmetric block cipher IDEA [19] wherein they used residue codes for fault detection in adders, multipliers, and XORs. 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. A simple encoding scheme of setting several bits of the message to a particular fixed value, 0 or 1, was proposed in [20]. 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 is targeted to detect faults in the transmission channel; encrypted data is transmitted and fault is detected after decryption at the receiver.

In this paper, we investigate systematic approaches to low cost, low latency CED of symmetric block ciphers. 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. The proposed techniques offer optimum trade-off between area and time overhead, performance, and fault detection latency. Further, they require minimal modification to the encryption device (encryption and decryption modules are unaffected) and are easily applicable to most of the symmetric block ciphers.

3. Symmetric block ciphers

A complete architecture of symmetric block cipher contains a key expansion module an encryption, 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 cipher encrypts (decrypts) the plain (cipher) text and generates 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, as shown in Figure 1. Exceptions include cipher with slightly different first and last rounds and ciphers with pre and post processing steps. A round of encryption (decryption)

performs a series of operations on the input data block and the round key(s) to generate the intermediate output data block. Output data block is then used as input data block for the next round. After a pre-determined number of rounds, and (if applicable) pre and post processing steps, cipher (plain) text is generated.

Decryption is the inverse of encryption. The inverse relationship exists at three levels. First level is at the operation level - encryption and decryption use mutually inverse operations, such as addition-subtraction, left rotation-right rotation, XOR-XOR etc. For example, a cipher text that is encrypted by left rotating the plain text Y bits can be right rotated by Y bits to recover the plain text. Inverse relationship between symmetric encryption and decryption extends to the round level. Consider an encryption round that adds X to the plain text, left rotates the result by Y bits, and then XORs with the key Z. Its corresponding decryption round must XOR the cipher text with key Z, right rotates by Y bits and then subtracts X. Finally, the inverse relationship between encryption and decryption exists at the algorithm level as well. Encryption algorithm consists of several encryption rounds with sequence number from 1 to R and the i^{th} encryption round is the inverse of $(R-i+1)^{th}$ decryption round. Plain text processed through encryption and decryption successively is recovered. We will discuss Advanced Encryption Standard (AES) finalist 128-bit symmetric block ciphers within this framework.

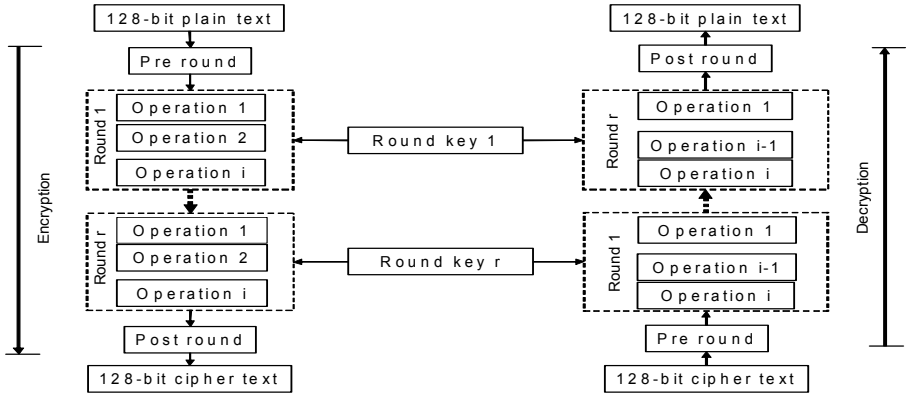

Figure 1: 128 -bit symmetric block cipher (a) encryption (b) decryption

RC6 symmetric block cipher encryption supports 20 rounds with each round using two round keys [21]. Four additional keys are used during pre and post processing. Each round of encryption uses two multiplications, two additions, two XOR operations, two fixed rotations and two variable rotations.

Rijndael supports multiple key lengths (128, 192, or 256 bits) and multiple block sizes (128, 192 or 256 bits) [22]. A 128-bit block, 128-bit key Rijndael encryption supports 10 rounds, with each round

using one round key. An additional key is used during pre-processing. Intuitively, Rijndael operates on a two-dimensional table of plain text bytes called 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 State (shift row), GF (2^8) multiplication with a constant of every column of State (mix column), and exclusive-or of round key with State (key-xor).

Serpent is a 128-bit block cipher that uses a 256-bit key [23]. Serpent encryption consists of 32 rounds and pre and post processing. Each round uses one round key (except for the last round, which uses two round keys). Exclusive-or (XOR) 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 encryption.

MARS supports a variable key size ranging from 128 to over 400 bits [25]. MARS encryption consists of 32 rounds - 8 forward mixing rounds, 16 main keyed transformation rounds each using two round keys and 8 backward mixing rounds. During pre and post processing four round keys are added to the input data.

Twofish cipher supports variable length key up to 256 bits [24]. The cipher comprises of 16 rounds with each round using 2 round keys. Four keys each are used during pre and post processing. Operations in an encryption round are key-dependent non-linear substitutions, GF (2^8) matrix multiplication, mixing operation based on pseudo-Hadamard transformation, key addition and bit wise rotation. Compared to other AES block ciphers Twofish satisfies the involution property resulting in identical data paths for both encryption and decryption. A function 'f' satisfies involution property if a = f (b) and b = f (a). Due to the common encryption and decryption data path CED architectures of Twofish are different from those of other block ciphers. A more detailed description is provided in the next section.

Table 1 summarizes various operations used by these 128-bit symmetric block ciphers in their encryption rounds. We have tried to present the operations in the order they are used within an encryption round. In some ciphers the order of the operations and the order of the keys in decryption is the exact inverse of that in encryption. Similarly, in some ciphers, such as Rijndael and Serpent, operations used in decryption are inverses of the corresponding encryption operations. Detailed description of each of these algorithms can be found in [21, 22, 23, 24, 25].

RC6	$\times(\text{mod } 2^{32})$	5-bit rot.	XOR	Variable rot.	$+(\text{mod } 2^{32})$ (key)	

Rijndael	S-box	Fixed rot.	$\times(GF(2^8))$	$\oplus$ (key)		
Serpent	Key-XOR	S-box	Lin. Transf.			
Twofish	S-box	$\times(GF(2^8))$	$+(\text{mod } 2^{32})$	$+(\text{mod } 2^{32})$ (key)	XOR	1-bit rot.
MARS	S-box	$+,\times(\text{mod } 2^{32})$ (key)	XOR	Variable rot.	5-,13-bit rot.	$-(\text{mod } 2^{32})$

Table 1: Operations used by encryption of 128-bit symmetric block ciphers

4. Concurrent error detection of symmetric block ciphers

A complete encryption device consists of an encryption module, a decryption module, a key ram, an input port and an output port. 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 bit coding, Hamming coding etc [26]. Hamming code is able to correct one bit fault and detect multiple bits faults in the protected word. However, it involves more hardware than parity bit code that can detect only the odd number of faults in the protected code. Since we need only error detection capability, parity bit code is a better choice when considering the fault probability and area overhead tradeoffs. In this paper we do not address CED for key RAMs and results of FPGA implementations in section 5 will exclude the corresponding overheads. We propose systematic approaches to low-cost, low-latency concurrent error detection techniques for encryption and decryption data paths. We describe algorithm level, round level, and operation level CED techniques that exploit the inverse relationship properties at different levels of symmetric encryption algorithms.

We initially assume that the device operates in half duplex mode (i.e. when encryption is performed, decryption module is idle and vice versa) and exploit this feature during CED. As mentioned earlier, due to its involution property Twofish is an exception since it requires only a single data path to carry out encryption and decryption in half duplex mode. Since a symmetric block cipher 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 then extend the proposed techniques to full duplex mode (i.e. when encryption and decryption modules are simultaneously processing incoming data) by trading off throughput and CED capability. The trade off between CED capability and throughput is analyzed in a later section.

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, which is temporally stored in a register. The output of decryption is compared with the input plain text and an error signal is set and the faulty cipher text is suppressed when there is a mismatch. Algorithm level CED during decryption is similar. Area overhead includes an additional register to store the original input, a comparator module, and a few multiplexers at the inputs of encryption, decryption and comparator modules.

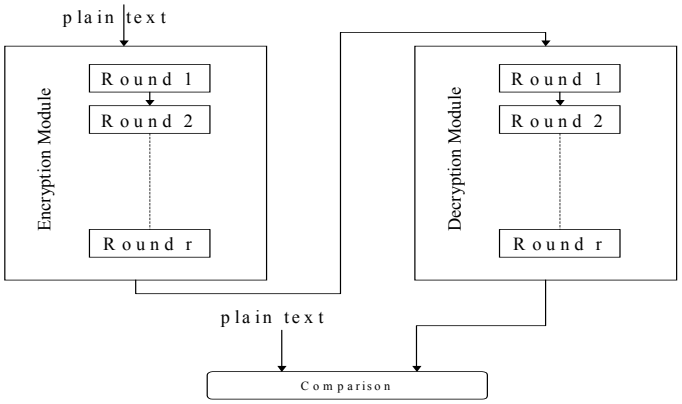

Figure 2 Encryption with algorithm level CED

Since round keys are stored in a key RAM subsequent blocks of plain text (cipher text) cannot be processed until both encryption and decryption of current plain text (cipher text) is finished (otherwise different round keys need to be accessed simultaneously from the RAM by encryption and decryption modules). Time overhead of encryption (decryption) for one block of data with algorithm level CED is twice 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, which is 100% time overhead. 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 RAMs - encryption key RAM and decryption key RAM, then 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 encryption (without CED) of one block of data.

Fault detection latency is the duration between the occurrence and detection of a fault. For a block cipher that has r rounds and n clock cycle(s) per round, the fault detection latency of algorithm level CED in the worst case is $2 \times n \times r$. Due to the involution property algorithm level CED for Twofish can detect only the transient faults since both normal and fault-detection computations use

same hardware.

4.2. Round Level CED

A closer look at the symmetric block ciphers reveals that the inverse relationship between encryption and decryption exists at the round level as well. **Any input data that is passed successively through one encryption round and the corresponding decryption round is recovered.** For almost all the symmetric block cipher algorithms, 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, fault-detection computations can also be performed at the round level. At the beginning of each encryption round input data is stored in a register before being fed to the round module. After one round of encryption is finished output is fed to the corresponding round of decryption. Output of the decryption round is then compared with the input data that was saved previously. If there is a mismatch, encryption process is halted and an error is signaled. Encryption with round level CED is shown in Figure 3.

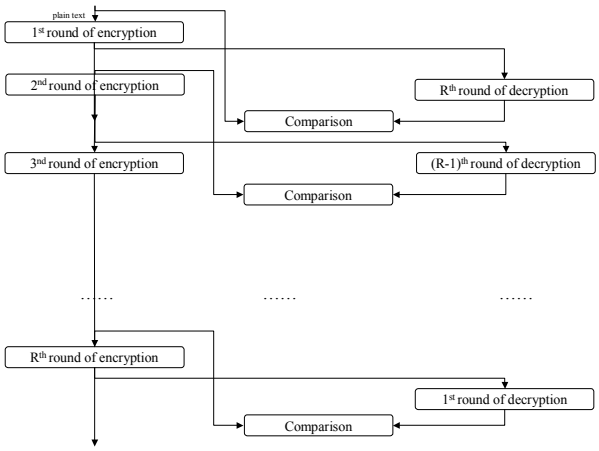

Figure 3 Encryption with round level CED

Performance penalty for encrypting one block of data with round level CED is only one 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 $2\times n$. Area overhead of round level CED is due to the additional registers, comparators, multiplexers and complex controller. Table 2 shows rounds of encryption and corresponding rounds of decryption for each of the AES finalist symmetric block ciphers. For example a 32-round implementation of Serpent round i of encryption (decryption) corresponds to round (33-i) of decryption (encryption). Similarly, MARS

contains 8 forward mixing, 16 keyed transformation and 8 backward mixing rounds. Hence, the i^{th} forward mixing round of encryption corresponds to the $(9-i)^{th}$ backward mixing round of decryption.

Since Twofish has a single data path its round level CED is implemented as follows. At the beginning of each encryption round input data is stored in a register before it is fed to the round module. After a round of encryption is over output is fed to the same round module for decryption (to perform CED). Output of decryption is then compared with the input data that was saved previously. Since there is only one round module that carries both encryption and decryption (for CED), following round of encryption has to be stalled until the decryption (for CED) of current round of encryption is finished. This results in 100% performance penalty and can only detect transient faults. Fault detection latency in this case is the time taken by one round of encryption or decryption.

RC6		MARS		Serpent		Twofish		Rijndael	
Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.
Pre Whitening	Post Blackening	Forward mixing round i	Backward mixing round 9-i	round i	round 33 – i	round i	round 17-i	round i	round 11 -i
Whitening round i	Blackening round 21– i	Keyed trans. round i	Keyed trans. round 17-i						
Post Whitening	Pre Blackening	Backward mixing round i	Forward mixing round 9-i						

Table 2: Encryption and corresponding decryption rounds that satisfy the inverse relationship

4.3. Operation level CED

Depending on the block cipher and its hardware implementation, each round may consume multiple clock cycles. Each 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, **applying input data through an encryption operation and the corresponding decryption inverse-operation yields the original input data**. This is shown in Figure 4.

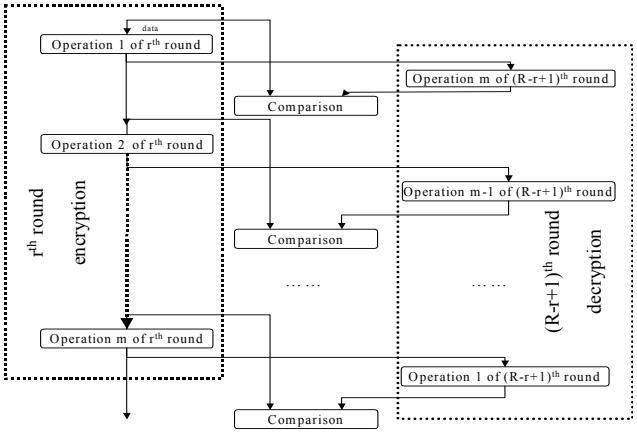

Figure 4: Encryption with operation level CED

Dotted boundary on the left shows the r^{th} encryption round while dotted boundary on the right shows the $(R-r+1)^{th}$ 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} (that is, last) operation of the decryption round. Output from operation 1 of encryption is fed into the corresponding (inverse) operation m of decryption. Such an operation level CED further improves the fault detection latency. In addition, it localizes the fault to the hardware implementing the operation. On the other hand, complexity of the design increases, as additional multiplexers, registers and comparators are required. Also, larger delay due to the additional multiplexers incurs additional performance penalty.

Operation level inverse relationship between encryption and decryption for the AES symmetric block ciphers is summarized in Table 3. For Serpent and Rijndael, operations shown in the *Encryption* column are inverses of the operations shown in the *Decryption* column. For RC6 and MARS, first operation (Operation 1 in RC6 and E-Function in MARS) of encryption is identical to (and not the inverse of) the first operation of decryption. 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.

RC6		MARS		Twofish		Serpent		Rijndael	
Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
Operation 1	Operation 1	E-function	E-function	Operation 1	Operation 1	XOR	XOR	S-box	$S\text{-box}^{-1}$
Operation 2	Operation 2^{-1}	Operation 2	Operation 2^{-1}	Operation 2	Operation 2	S-box	$S\text{-box}^{-1}$	Shift row	Shift row^{-1}
						LT	LT^{-1}	Mix column	Mix column^{-1}
								Key Xor	Key Xor

Table 3: Encryption and corresponding decryption operations at round level

Operation 1 and Operation 2 of Twofish satisfy the involution property. Performance penalty is only one clock cycle since Operation 2 of encryption can be carried in parallel with the Operation 1 of decryption (for CED) and fault detection latency is 2 clock cycles.

4.4. Comparison of Algorithm, Round and Operation level CED approaches

Table 4 summarizes the performance and fault detection latency in terms of clock cycles for different CED architectures for AES and other symmetric block ciphers. 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). Similarly, one round of Serpent consumes two clock cycles. This translates into 64 clock cycles for a basic implementation of 32-round Serpent encryption (32×2). Number of clock cycles for CED architectures shown in Table 4 has been computed similarly. Duration of clock cycle varies across different ciphers as well as different architectures of the same cipher.

Algorithm	No CED	algorithm-level CED		round-level CED		Operation-level CED	
	# of cycles	# of cycles	Detect. Latency	# of cycles	Detect. Latency	# of cycles	Detect. Latency
RC6	42	84	84	44	4	43	2
Serpent	64	128	128	66	4	65	2
Twofish	34	68	68	68	4	35	2
Rijndael	44	88	88	48	8	45	2

Table 4: Comparison of AES finalist 128-bit symmetric block ciphers

4.5. Discussion

Straightforward duplication of the encryption and decryption data paths results in two copies of encryptions hardware and two copies of decryption hardware and is suitable for encryption devices operating in full duplex mode. Basic idea of our approaches is that given an encryption device that operates in half duplex mode of operation we utilize the decryption data path to validate the encryption data path and vice versa. Since both encryption and decryption data paths are busy when encrypting or decrypting one block of data, the proposed approaches are suitable for devices that operate in half duplex mode.

Since all successive inputs plain text are encrypted and then decrypted to ensure fault-free operation, maximum CED capability is achieved and the check frequency is 100%. Check frequency is defined as the ratio between the number of plain text that goes through both encryption and decryption (for CED) and the number of the total processed plain texts. Check frequency can be

reduced to enable full duplex mode operation. For example, if only every fourth input plain texts goes through both encryption and decryption modules device operates in full duplex mode 75% of the time. Check frequency can be tuned to trade-off throughput and reliability as well as security.

For a design with CED capability, faults in the additional logic added for CED also affects the performance of crypto-devices. More the area overhead, more vulnerable the device is. Our approaches reduce the area overhead by utilizing the existing encryption and decryption data path, thereby minimizing the faults due to the additional logic required for CED. We assume the comparator to be self-checking.

Symmetric encryption algorithms can operate in multiple modes such as Electronic Code Book (ECB), Cipher Block Chaining (CBC) and Output Feed Back (OFB). In the ECB mode, plain text is used directly as input data block to the block cipher. In other words, each block of plain text has a corresponding cipher text value and vice versa. In the CBC mode, first block of plain text is exclusive-ored with an initialization vector (IV) before encryption. Then, the first block of cipher text is exclusive-ored with the next block of plain text to form the next input data block and so on. In the OFB mode plain text is divided into data blocks of K bits each. An IV of length L is placed in the least significant bits of the input data block with the unused bits set to 0. Input block is processed through encryption module to produce an output data block. Exclusive-oring a K-bit plain text data block with the most significant K bits of the output data block produces the cipher text. Discarding K bits of the old IV and concatenating the old plain text data block to the result derives the new IV. These block cipher modes can be categorized into feedback and non-feedback modes. In the feedback mode (CBC, OFB) successive input blocks cannot be processed until the encryption/decryption of the current data block is finished. In the non-feedback mode (ECB) input data blocks are independent of the current and subsequent output data blocks. Pipelining can be applied in the non-feedback mode to improve the throughput at the cost of increasing the hardware. Since these operation modes do not affect the data flow of encryption /decryption, our approaches can be used with encryption algorithms operating in any of these modes. For example, encryption with the proposed CED schemes in CBC mode can be implemented by exclusive-oring the incoming plain text and the current cipher text to form the next input block without affecting the internal CED structure. We implemented AES finalists in the ECB mode without pipelining and the results are presented in the next section.

5. FPGA Implementation based validation

Validation of the proposed CED techniques was carried out by implementing the 128-bit

symmetric block ciphers with different CED mechanisms on Xilinx FPGA device, XCV1000BG560-6. All CED architectures were modeled in VHDL and functionally verified using Modeltech's Modelsim VHDL simulator. Synplicity's Synplify was used for synthesis and Xilinx Foundation PAR tool was used for mapping and place and route. Due to the resource limitations we did not implement MARS. Similarly, due to the large performance penalty we did not implement the round level CED scheme for Twofish. Results of our various FPGA implementations 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. Throughput of a cipher represents the total number of bits encrypted per second and is calculated as the number of bits encrypted / (operation cycles * clock duration). Performance degradation is obtained as: 1 – (throughput of CED architecture/ throughput of basic architecture).

		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	N/A	3467
	Serpent	8073	9376	9659	9974
Maximum frequency (MHz).	**RIJNDAEL**	46.93	36.44	37.60	36.69
	RC6	23.99	21.76	20.740	16.87
	Twofish	20.16	18.98	N/A	19.072
	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	N/A	67.80
	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	N/A	6.28
	Serpent		16.14	19.15	23.55
Performance Degradation (%)	**RIJNDAEL**	-	61.15	26.55	23.56
	RC6	-	54.64	17.48	31.31
	Twofish	-	52.92	N/A	10.67
	Serpent		46.98	11.05	8.01

Table 5: Summary of FPGA implementation of CED architectures

From Table 5 it can be seen that the clock frequency and detection latency decrease, while the area overhead increases, as the granularity of CED increases. **Decrease in fault detection**

latency/increase in area and **decrease in fault detection latency/decrease in throughput** ratios are more significant between algorithm level CEDs and round level CEDs than they are between round level CEDs and operation level CEDs. Maximum performance degradation is 31.31% while the maximum area overhead is 39.20%. Main area overhead of our approaches is due to the additional multiplexers. FPGA implementation of multiplex is not efficient and is equivalent to an adder with the same bit width. This increases the overhead associated with our approaches. In addition these are the results for un-optimized, reference implementations.

6. CED Case Study: 128-bit Rijndael block cipher

Since Rijndael has been chosen as the AES, we use it as our case study. 128-bit Rijndael symmetric block cipher uses 11 rounds each for encryption and decryption. A decryption round is realized by applying the inverse of the operations used in the encryption round (s-box, shift-row, mix-column, and key addition) in the reverse order. Consequently, the sequence of operations in a decryption round is: key-subtraction → inverse mix column → inverse shift-row → inverse s-box. In our AES implementation one round of encryption (decryption) consumes four clock cycles. Therefore, encrypting (decrypting) one 128-bit block of plain (cipher) text consumes 11 x 4 = 44 clock cycles. Round keys are stored in a register file for use during encryption and decryption.

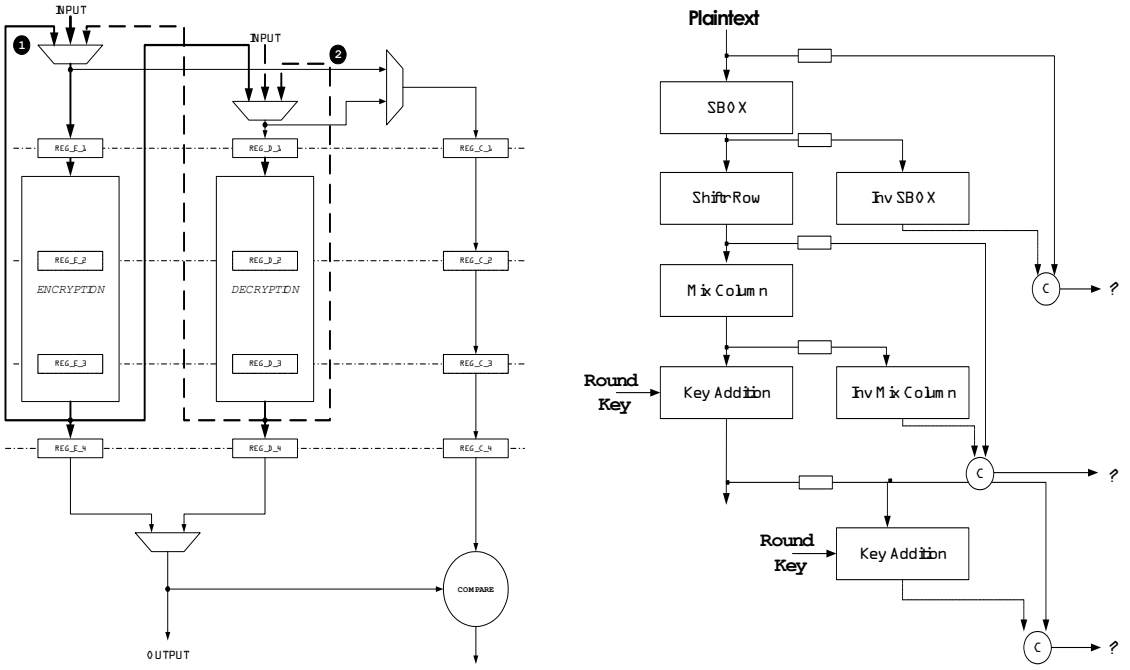

Figure 5: Rijndael CED architectures: (a) round-level and (b) operation level

Incorporation of algorithm level CED into Rijndael is straightforward. Since we already have individual encryption and decryption modules, we connect the output of encryption (decryption)

module to the input of decryption (encryption) module via a *2 ×1* mutiplexer. Area overhead includes comparator to detect mismatch, registers to store the input, output and intermediate data blocks, and multiplexers to control data flow. Fault detection latency using this approach is 44×2 = 88 clock cycles.

Figure 5 (a) shows the round-level CED architecture of Rijndael. Thin, dotted horizontal lines denote clock cycle boundaries, while thick, solid and dotted lines denote flow of plain and cipher text respectively. Output of encryption-round component is fed to the decryption-round component after each round of processing. Output of decryption-round component is then compared to the input to the encryption round using a comparator. The comparator generates an error signal if there is a mismatch. Since the comparison is executed at the end of each decryption round, fault detection latency is reduced to 8 clock cycles (4 cycles each for a round of encryption and decryption). Clock duration increases from 21.3ns to 27.74ns since comparator added delay in the critical path.

Finally, the inverse relationships that exist at the operation level can be exploited for implementing CED with lowest fault detection latency. Figure 6 shows the CED architecture for s-box and inverse s-box pair of Rijndael. Identical architectures are used for other encryption and decryption operations. Fault detection latency for the operation level CED is reduced to two clock cycles. Additional benefit of operation level CED is that it can also identify the faulty operation module pair.

Table 5 shows that algorithm-level CED and round-level CED have comparable area overheads (approximately 20%) while operation-level CED has approximately 38% area overhead. Performance degradation of the algorithm-level CED is 61% while round-level and operation-level CED has comparable performance degradation (25%).

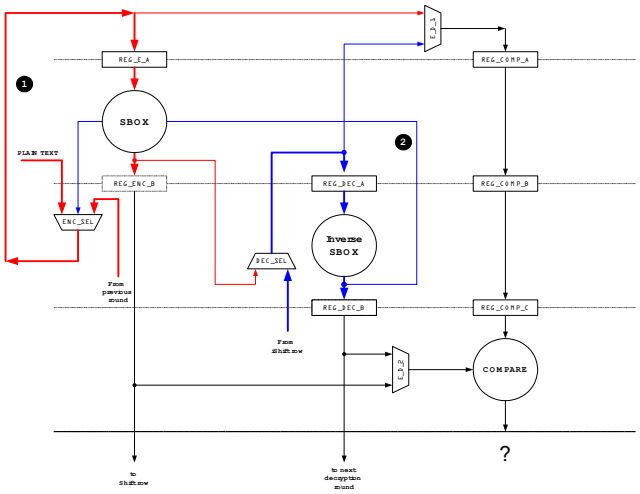

Figure 6: Block diagram for operation level CED of (s-box, inverse s-box) pair

7. Conclusions

We have presented algorithm level, round level and operation level CED architectures for symmetric block ciphers. Our techniques are algorithm-independent and can be applied to almost any symmetric block cipher. 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. 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. E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Crytptosystems", *Journal of Cryptography*, Vol. 4, No. 1, 1991, pp. 3-72.
2. M. Matsui, "Linear Cryptanalysis Method for DES Cipher", *Proceedings of Advances in Cryptology-Eurocryp*, Springer-Verlag, 1994, pp. 386-397.
3. J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side-Channel Cryptanalysis of Product Ciphers", *Proceedings of ESORICS*, Springer-Verlag, September 1998, pp. 97-110.
4. C. Harpes and J. Massey, "Partitioning Cryptanalysis", *Fast Software Encryption, 4th International Workshop Proceedings*, Springer-Verlag, 1997, pp. 13-27.
5. S. Vaudenay, "An experiment on DES Statistical Cryptanalysis", *3rd ACM Conference on Computer and Communications Security*, ACM Press, 1996, pp. 139-147.
6. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", *Proceedings of Advances in Cryptology-CRYPTO*, Springer-Verlag, 1996, pp. 104-113.
7. 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.
8. P. Kocher, J. Jaffe, B. Jun, "Introduction to Differential Power Analysis and Related Attacks". http://www.cryptography.com/dpa/technical, 1998.
9. W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk", *Computers and Security*, Vol. 4, 1985, pp. 269-286.
10. Erhart Moller, "Protective Measures Against Compromising Electro Magnetic Radiation Emitted by Video Display Terminals", *Phrack Magazine*, Vol. 4, Issue 44, File 10 of 27.
11. 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, Vol. 1233, 1997, pp. 37-51.
12. A. K. Lenstra, E. R. Verheul, "Selecting Cryptographic Key Sizes", *Public Key Cryptography Conference*. http://www.cryptosavvy.com/cryptosizes.pdf
13. 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
14. 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, Paris, April 7-9, Springer LNCS 1361, 1997, pp. 125-136.

15. E. Biham, A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems", *Proceedings of Crypto*, 1997.
16. 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
17. R. Anderson, M. Kuhn, "Low cost attack on tamper resistant devices", *5th International Workshop on* Security Protocols, Paris, Springer LNCS 1361, 1997.

http://www.cl.cam.ac.uk/ftp/users/rja14/tamper2.ps.gz

18. 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.
19. 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*, 1995, Vol.1, pp. 397-400.
20. S. Fernandez-Gomez, J. J. Rodriguez-Andina, E. Mandado, "Concurrent Error Detection in Block Ciphers", *IEEE International Test Conference*, 2000.
21. Ronald L. Rivest, M. J. B. Robshaw, R. Sidney, Y. L. Yin, "The RC6™ block cipher", ftp://ftp.rsasecurity.com/pub/rsalabs/aes/rc6v11.pdf
22. J. Daemen, V. Rijmen, "AES proposal: Rijndael", http://www.esat.kuleuven.ac.be/~rijmen/rijndael/ rijndaeldocV2.zip
23. 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
24. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, "Twofish: A 128-bit cipher", http://www.counterpane.com/twofish.pdf
25. C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford, N. Zunic, "MARS - A candidate cipher for AES", http://www.research.ibm.com/security/mars.pdf
26. M. Abramovici, M. Breuer, A. Friedman, "Digital Systems Testing and Testable Design", IEEE Press Marketing, ISBN 0-7803-1062-4.