Fault-Resilient Lightweight Cryptographic Block Ciphers for Secure Embedded Systems

Mehran Mozaffari-Kermani, *Member, IEEE*, Kai Tian, Reza Azarderakhsh, *Member, IEEE*, and Siavash Bayat-Sarmadi, *Member, IEEE*

***Abstract*—The development of extremely-constrained embedded systems having sensitive nodes such as RFID tags and nanosensors necessitates the use of lightweight block ciphers. Nevertheless, providing the required security properties does not guarantee their reliability and hardware assurance when the architectures are prone to natural and malicious faults. In this letter, error detection schemes for lightweight block ciphers are proposed with the case study of XTEA (eXtended TEA). Lightweight block ciphers such as XTEA, PRESENT, SIMON, and the like might be better suited for low-resource deeply-embedded systems compared to the Advanced Encryption Standard. Three different error detection approaches are presented and according to our fault-injection simulations, high error coverage is achieved. Finally, field-programmable gate array (FPGA) implementations of these proposed error detection structures are presented to assess their efficiency and overhead. The schemes presented can also be applied to lightweight hash functions with similar structures, making the presented schemes suitable for providing reliability to their lightweight security-constrained hardware implementations.**

***Index Terms*—Cryptography, error detection, security.**

I. Introduction

ONE defining trend of this century's IT landscape is the extensive deployment of tiny computing devices such as radio frequency identification (RFID) devices and wireless nanosensor nodes. The sensitivity of such applications makes the algorithms such as the advanced encryption standard (AES) [1] and lightweight cryptography essential to reach acceptable confidentiality. For instance, for the tiny encryption algorithm (TEA), a new extended variant, XTEA, was developed [2]. This algorithm is notable for its simplicity (making it very suitable for hardware implementations) and is used widely in providing lightweight security.

Manuscript received September 16, 2014; accepted October 22, 2014. Date of publication October 24, 2014; date of current version November 20, 2014. This work was supported in part by Texas Instruments Faculty Award, granted to M. Mozaffari-Kermani and R. Azarderakhsh (K. Tian is their joint graduate student under this award). This manuscript was recommended for publication by T. Eisenbarth.

M. Mozaffari-Kermani and K. Tian are with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: m.mozaffari@rit.edu; txk1844@rit.edu).

R. Azarderakhsh is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: rxaeec@rit.edu).

S. Bayat-Sarmadi is with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (e-mail: sbayat@sharif.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2014.2365099

An important reason leading to natural faults in the very-large-scale integration (VLSI) implementations is hardware failures. In cryptographic hardware and embedded systems, the adverse effects of such faults are amplified considering not only the sensitivity of such structures but the possibility of mounting active side-channel analysis attacks, commonly referred to as fault attacks. For cryptographic architectures such as the AES, much research has been carried out to achieve various concurrent error detection (CED) structures, see, for instance, [3]–[10] (also, refer to [11] for reliable architectures for lightweight cryptography).

To date, many research works have analyzed lightweight block ciphers such as the recent one on SIMON [12]. As one of the fastest and most efficient block ciphers in existence, XTEA is used for some real-life cryptographic applications. This block cipher only uses simple addition, XOR, and shift functions, and has a very small code size. This makes XTEA an excellent candidate to provide confidentiality for nodes having limited memory and computational power.

In this letter, using this algorithm as the case study, the reliability and fault resilience of lightweight crypto-architectures are assessed. We note that this choice does not confine the proposed methods for other lightweight cryptographic algorithms (such as SIMON). The high error coverage of the presented schemes would meaningfully increase the difficulty for potential fault attackers. We use signature-based and recomputing with rotated operands (RERO) approaches based on the objectives of protection, reliability requirements, and overheads to be tolerated. Through error simulations, it is shown that with high error coverage, reliable architectures are devised. Furthermore, through field-programmable gate array (FPGA) implementations on Virtex-5 FPGA devices, the overheads of the presented architectures are shown to be acceptable for resource-constrained applications.

II. Preliminaries

The lightweight block cipher XTEA accepts a cryptographic key of 128 bits and a 64-bit block size. The input block is separated into two halves X and Y. These are applied to a Feistel network for N cycles and N is normally 32.

Algorithm 1 shows the process through which this algorithm derives the ciphertext. Within XTEA, all additions and subtractions are modulo 2^{32}. Logical left shifts by 4 bits are denoted as $\ll 4$ and logical right shift by 5 bits are denoted as $\gg 5$. The XOR function is denoted as "$\oplus$" in this algorithm. The first part of the algorithm is a permutation function and the second part is a subkey generation function. The 128-bit user key can be split

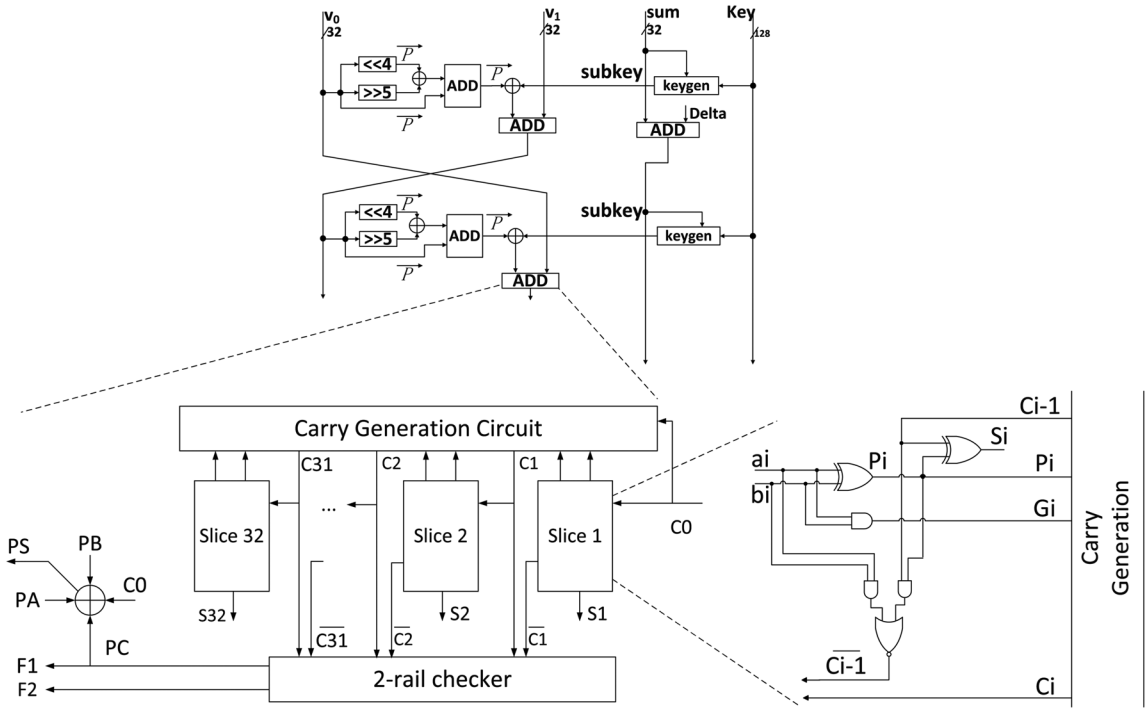

Fig. 1. Carry checking/parity prediction adder avoiding duplication of the carry generation block.

into four blocks. Each block is a 32-bit subkey. The function $key[sum]$ chooses one block out of the four subkey depending on 1st and 0th bits (12th and 11th bits in the second half cycle) of sum. The XOR function is applied to the result of the permutation function and the subkey generation function and, then, this result is applied to x and y by addition when encrypting or subtraction when decrypting. A whole cycle can be split into two half cycles. In each half cycle, a new value for v_0 and v_1 is computed. Between the first and the second half cycle, a new value for sum is computed. sum increases by a constant $Delta$, which equals to the integral part $(\sqrt{5}-1)\times 2^{31}$ in hexadecimal form. This addition can be included in the first half cycle.

Algorithm 1 The XTEA encryption algorithm.

Inputs: 64-bit data: $v[0]$-$v[1]$, 128-bit key: $key[0]$-$key[3]$.
Initialize (encryption): $v_0 = v[0]$, $v_1 = v[1]$, $sum = 0$, $delta = 0x9E3779B9$, n: number of rounds.
1. for $i = 0$ to $n-1$ do
2. $v_0 \leftarrow v_0 + (((v_1 \ll 4) \oplus (v_1 \gg 5)) + v_1) \oplus (sum + key[sum\&3])$.
3. $sum \leftarrow sum + delta$.
4. $v_1 \leftarrow v_1 + (((v_0 \ll 4) \oplus (v_0 \gg 5)) + v_0) \oplus (sum + key[sum \gg 11\&3])$.
5. end for.

III. Proposed Error Detection Schemes

In this section, two different error detection approaches are proposed with the case study of XTEA. We would like to emphasize that although we use this case study, the proposed methods are applicable with slight modifications to similar lightweight block ciphers (such as SIMON) and hash functions. For the sake of brevity, only the schemes for the encryption process are presented.

A. Signature-Based Diagnosis

For the XOR and shift functions, the parity prediction function ($\hat{P}$ is used for predicted parities) is straightforward. For instance for XOR $\hat{P}_S = \sum_{i=0}^{n-1}(a_i \oplus b_i) = \sum_{i=0}^{n-1} a_i \oplus \sum_{i=0}^{n-1} b_i = P_A \oplus P_B$, assuming two inputs are A and B (with bits a_i and b_i) and the actual parities for inputs are P_A and P_B.

Assuming the two inputs are A and B, then, the output bit s_i is equal to $a_i \oplus b_i \oplus c_{i-1}$. Considering modulo-2 addition, the output parity is given by the following expression $\hat{P}_S = \sum_{i=0}^{n-1}(a_i \oplus b_i \oplus c_{i-1}) = \sum_{i=0}^{n-1} a_i \oplus \sum_{i=0}^{n-1} b_i \oplus \sum_{i=0}^{n-1} c_{i-1} = P_A \oplus P_B \oplus \sum_{i=0}^{n-1} c_{i-1}$, where c_{i-1} is the carry input of the ith bit.

In a system using parity encoded data, the parities P_A and P_B are already available. Thus, only $P_C = \sum_{i=0}^{n-1} c_{i-1}$ has to be computed to predict the parity code.

An advanced architecture of carry checking/parity prediction adders is through removing the parity generator, avoiding duplication of complex carry generation blocks, and using partial carry duplication [13]. Fig. 1 shows the carry checking/parity prediction adder avoiding duplication of the carry generation block and the architecture of adder bit slice using partial carry duplication.

F1 and F2 in Fig. 1 are two outputs of the double-rail checker (these are used for codeword verifications). Because of the property of double-rail checkers that they have a one-to-one correspondence with the parity trees, they can be viewed as parity generators with double-rail inputs and outputs. The inputs to the double-rail checker are C_i and $\bar{C}_i$, so the output F1 is P_C. P_S

can be obtained by making P_A, P_B, P_C, and C_0 going through an XOR gate.

To avoid duplication of the carry generation block, the structure of slice in adder should be changed as shown in Fig. 1. The following solutions were used to solve this problem: using the same carry generation logic in the ripple-carry adder to guarantee low hardware cost and using the carry inputs from the normal carry generation logic rather than the previous slice of the check carries to guarantee high speed.

The entire parity prediction schemes for XTEA can be reached by combining these three parity prediction approaches.

B. Recomputing with Rotated Operands (RERO)

Error detection schemes based on time redundancy often suffer from the inability to detect permanent faults. However, RERO is capable of detecting not only the transient faults which are common in fault attacks but also the permanent faults affecting logic gates in the cryptographic hardware and embedded systems. Suppose ψ and ψ^{-1} are n-bit rotations (or cyclic shifts) toward the least and most significant bits of a binary operand, respectively. Moreover, let π be the input to an arithmetic function g, and $g(\pi)$ be its output in such a way that $\psi^{-1}(g(\psi(\pi))) = g(\pi)$. To apply the RERO method, we need to store the result of the first computation (first run) and compare it against the result of the second computation (second run). If the results are different, it indicates an error is alerted by the error indication flag.

The most involved part for RERO is the one for addition. The correctness of carry-in for logic $i+1$th bit and carry-out from logic $n-1$th bit should be guaranteed. As such, one extra bit is added as the most significant bit (using an $n+1$ adder) and the logic pattern before rotation becomes

$$@n-1n-2\ldots i+1i\ldots 210$$

where @ is the added bit which is stuck-at zero. The logic pattern after rotation becomes

$$i\ldots 210@n-1n-2\ldots i+1,$$

The values of@is always set to "0" even if it is not actually derived as "0". Thus, logic $n-1$th bit is not able to affect logic 0th bit in the rotated operation as well as the normal one.

Moreover, one needs to link the carry-out from the most significant bit to the carry-in of the 0th bit. Due to the fact that the values of the extra bit are always "0", the carry-out from the most significant bit is "0" in the first computation and it will not affect 0th bit. During the second computation, the most significant bit contains logic ith bit. The carry-out from logic ith bit is applied to the carry-in of logic $i+1$th bit.

Throughput and Efficiency Considerations: Time redundancy techniques suffer from degradations in performance. However, it is possible to increase the frequency of computations (and thus to increase the efficiency and throughput) through subpipelining. Suppose one pipeline-register has been placed to subpipeline the structures. Let us denote the two halves of pipelined stages by Π_1 and Π_2. The original input is first applied to the architecture in the first cycle. In the second cycle, while the second half of the circuit (Π_2) executes this first input, the rotated variant of the first input is fed to the first half of the circuit (Π_1). This trend is consecutively executed until the last rotated input is derived. As such, we can take advantage of multilevel subpipelining to reduce the throughput degradation of the proposed scheme. Although the added subpipelining registers slightly increase the induced hardware overhead, it is more preferable to use the time-redundancy schemes which introduce much more overall design overhead.

IV. Error Simulations and FPGA Implementations

To evaluate the error detection capability of the proposed structures, fault-injection simulations have been performed. The fault model used is elaborated in the following.

A. Fault Model

Throughout this letter, both single and multiple stuck-at faults have been considered (note that these could be transient or permanent). These two models cover both malicious fault attacks and natural faults. Indeed, single stuck-at faults model the natural failures (such as single event upsets) and are the ideal cases for the attackers. However, due to technological constraints, single stuck-at fault injections become more difficult for an attacker to gain information (it is still a possibility due to larger components such as bus-lines). Thus, multiple bits will actually be flipped, and, thus, multiple stuck-at faults are also considered in this letter.

B. Simulation and Implementation Results

If exactly only one bit error appears, the coverage is 100%, thus, no simulation is needed. Most internal faults can be modeled by transient random faults. These could be localized faults with 50% detection rate if they affect just one parity unit or very high if randomly-distributed faults occur. We note that each parity cannot detect with more than 50% rate but the combination of parity bits if the fault model is multiple, randomly-distributed faults (permanent or transient) can detect with higher rate. One side-note is that through testing, permanent faults can be detected as well but transient faults (in case they are randomly-distributed) can be detected with high ratio using the proposed methods. We note that, however, even a half-round has a number of parities so theoretically, the detection rate is much higher even for half-round localized faults affecting the blocks covered by the respective parities. 10 000 faults are injected using eight different test cases and validated for error coverage and assessed through a linear-feedback shift register (LFSR)-based simulation environment. It is noted that we use Fibonacci implementation LFSRs with the required output taps for injecting random multiple errors, where the numbers, locations, and types of the errors are randomly chosen.

For each injection, error indication flags are monitored, and the detected errors are counted. The results of the performed simulations show very high error coverage (all the cases have at least 9 996 faults detected out of 10 000 samples, i.e., 99.96% detection rate for this case). For the signature-based scheme, multiple parities for multiple detection points (shown in Fig. 1) are used to achieve such high coverage as single parities are ineffective even for single errors. It is noted that the implemented

TABLE I
PERFORMANCE DEGRADATION COMPARISON

Structure	Delay (ns)	Overhead	Throu. (Mbps)	Deg.
Original	3.833	-	39.57	-
RERO[1]	2.891	N/A	26.42	33.2%

[1] One stage subpipelined architecture.

architecture utilizes the RERO method for one computation of a round to have a practical scheme.

If an attacker is capable of injecting faults in the parity circuitry, the following cases can happen: 1). the injected single or multiple stuck-at faults (if not masked) in only the parity prediction circuitry are detected; 2). for multiple stuck-at faults, in both the original and prediction circuits (which are not ideal cases for attacks), such dual injections can make the respective parity prediction blocks ineffective (if not masked). However, it is emphasized that multiple stuck-at faults are not preferable and even if they occur, it will make just the respective parity blocks ineffective and might not have large effects on final error coverage. Nevertheless, the presented RERO scheme is not vulnerable to such injections and its comparison unit is assumed to be fault tolerant.

In this section, we also present the results of the overhead assessments using the FPGA hardware platforms. The analysis has been performed for the original and the error detection structures of the encryption process of the XTEA. Vivado version 2013.2 and Kintex FPGA device xc7k70tlfbg676-2L have been utilized for the FPGA implementations. VHDL has been used as the design entry for the original and the error detection structures.

To benchmark the performance of the proposed schemes, we have done implementations for the original and fault diagnosis scheme using RERO as seen in Table I (this scheme is chosen for implementations as it is suitable for low-area embedded architectures). Moreover, based on the subpipelining approach we presented in this letter, one can alleviate the inherent performance degradations of the RERO method. Specifically, with the expense of adding registers for deep subpipelining (for instance, one stage subpipelining in Table I), higher frequencies are achieved for the RERO scheme which make the degradations in throughput less intense.

In Table II, the area and power consumptions for the original and the error detection structures of the XTEA are presented (working frequency of 100 MHz). We have performed implementations for the hardware redundancy scheme and with 333 slices used (area overhead of 88.1%), this scheme is not preferable for low-complexity architectures. As seen in Table II, the area and power consumption overheads of the RERO structure are shown (we get around 5% increase in power consumption). Based on the simulation results, these overheads are added for the error coverage of very close to 100%. The proposed fault diagnosis approaches provide high error coverage at the expense of the acceptable overheads on the FPGA hardware platforms, making the hardware architectures of the XTEA more reliable.

TABLE II
AREA AND POWER CONSUMPTION OVERHEADS COMPARISON

Structure	Area (#slices)	Overhead	Power (mW)	Overhead
Original	177	-	87	-
RERO	228	28.8%	91	4.6%

V. CONCLUSION

In this letter, two fault diagnosis approaches for the lightweight block cipher XTEA have been proposed. These include parity-based structure and RERO structure. The results of the simulations show very high error coverage (very close to 100%) for the presented error detection structures for the injected faults. Moreover, the FPGA implementation analysis results show acceptable overheads for the XTEA when the presented schemes are utilized. The proposed schemes can be used to protect the extremely-sensitive and resource-constrained applications.

REFERENCES

[1] National Institute of Standards, and Technologies, "Announcing the advanced encryption standard (AES)," in *Proc. Federal Inf. Processing Standards Pub.*, Nov. 2001, no. 197.
[2] D. Wheeler and R. Needham, "TEA extensions," in *Technical report*. Cambridge, U.K.: Cambridge Univ. Press, Oct. 1997.
[3] C. H. Yen and B. F. Wu, "Simple error detection methods for hardware implementation of advanced encryption standard," *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 720–731, Jun. 2006.
[4] T. G. Malkin, F. X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Proc. Int. Workshop FDTC*, Oct. 2006, pp. 159–172.
[5] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 608–622, May 2010.
[6] A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "High-performance concurrent error detection scheme for AES hardware," in *Proc. Int. Workshop Cryptographic Hardware Embed. Syst.*, Aug. 2008, pp. 100–112.
[7] P. Maistri and R. Leveugle, "Double-data-rate computation as a countermeasure against fault analysis," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1528–1539, Nov. 2008.
[8] G. Xiaofei and R. Karri, "Recomputing with permuted operands: A concurrent error detection approach," *IEEE Trans. Comput.-Aided Des.*, vol. 32, no. 10, pp. 1595–1608, Oct. 2013.
[9] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Reliable hardware architectures for the third-round SHA-3 finalist grostl benchmarked on FPGA platform," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (DFT)*, Oct. 2011, pp. 325–331.
[10] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A lightweight high-performance fault detection scheme for the advanced encryption standard using composite fields," *IEEE Trans. Very-Large Scale Integr. (VLSI) Sys.*, vol. 19, no. 1, pp. 85–91, 2011.
[11] M. Mozaffari Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5925–5932, Dec. 2013.
[12] A. Aysu, E. Gulcan, and P. Schaumont, "SIMON says: Break area records of block ciphers on FPGAs," *IEEE Embed. Syst. Lett.*, vol. 6, no. 2, pp. 37–40, Jun. 2014.
[13] M. Nicolaidis, "Carry checking/parity prediction adders and ALUs," *IEEE Trans. Very-Large Scale Integr. (VLSI) Sys.*, vol. 11, no. 1, pp. 121–128, 2003.