ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 10(19), DOI: 10.17485/ijst/2017/v10i19/111355, May 2017* ISSN (Online) : 0974-5645

High Performance and Reliable Fault Detection Scheme for the Secure Hash Algorithm

Fatma Kahri*, Hassen Mestiri, Belgacem Bouallegue and Mohsen Machhout

Department of Electronics, University of Monastir, Tunisia; Kahrifatma@gmail.com, hassen.mestiri@yahoo.fr, belgacem_bouallegue@yahoo.fr, machhout@yahoo.fr

Abstract

Background/Objectives: Among the powerful techniques against the protected hash functions (SHA) the method of fault injection attacks. For obtain the confidential information, the method consist to inject this attacks during the process of the hash algorithm. In literature, they proposed a many methods of countermeasures to secure the SHA implementation against these attacks. **Methods/Statistical Analysis:** In this paper, we proposed a new scheme of fault detection; this scheme is based on the combines of two redundancies named hybrid redundancy for the hash algorithm. The weaknesses and the strengths of our proposed method against the fault attacks are discussed. **Findings:** our schemes proposed reaches 99.999% fault coverage. Moreover, we are implemented our proposed scheme on Xilinx Virtex-5 and Virtex-II Pro FPGA. Its area overhead fault coverage, frequency and throughput degradation have been compared and it is shown that our proposed scheme allows a trade-off between the security of the SHA and hardware overhead. **Application/Improvements:** Compared to other work, our fault detection scheme has the important performances in terms of frequency, occupied slices, fault coverage and throughput.

Keywords: Countermeasures, Fault Detection Schemes, Fault Attacks, Security, Secure Hash Algorithm (SHA)

1. Introduction

Hash functions pertain public cryptographic functions. The principle of this algorithm is to obtain a fixed size from a message of variable size. The fixed size is used as the authenticator. Hash functions are used in many applications such as stocking the password, digital signature crypto processor, and verification. For ensuring a high security protocols, the hash functions are used for message integrity objective and entity identification.

In a cryptographic hash algorithm, first, an arbitrary length message of is padded. Second, it's the initial message is divided into n blocks, in order to obtain the message digest you must convert the input to output fixed length[1]. The padding output are calculated iteratively by a hash unite (compression function); the results are the message digest. The cryptographic hash function is actually implementation in very large of applications. Same, to secure the Secure Hash Algorithm (SHA) architecture against the fault injection attacks, such as the Simple Fault Analysis (SFA) and the Differential Fault Analysis (DFA). The idea is consist to pull out the secret key by the method of injection of faults into a crypto processor. After, we are analyzing the results from the fault presence of the system.

To describe the attacks by faults three essential phases are presented.

- First phase: Fault models vary in various aspects including the attack position, e.g., a memory or a register position, setting, or resetting, the attack time point, the number of attacked bits, and the kind of bit manipulation such as bit flipping, the duration of the fault.
- Second phase: we are defined the fault model previously which makes to inject the according errors. For this, various techniques can be used such as applying power supply peaks[2-10], magnetic fields[11], optical signals[12], temporary clock drifts[13-14], and high temperature[15].
- Third phase: to extract the secret-key of crypto processor the results is estimated in the light.

* *Author for correspondence*

In the literature a large number of fault detection schemes are proposed, also, different types of faults are presented such as permanent or transient.

In this work, we describe our proposed scheme for the implementations of hash functions. This paper is organized as follows. The related background knowledge is described in section 2. The error analysis simulations are presented in Section 3we are described and classified the hash functions implementations a according to the sort of the redundancies scheme in section 4. The proposed are compared in terms of capabilities implementation cost and fault detection. In section 5, the proposed fault detection scheme for the SHA implementation is existed in section 6. Its experimental synthesis results and performances are discussed and compared in terms of frequency, fault coverage, throughput, and area in section 7. Section 8 concludes the paper.

2. Background

The NIST was first published the Secure Hash Standard (SHA) in 1993[1617]. A new hash function SHA-2 was proposed in 2001. These new hash families SHA-2, use larger digest messages, making them more resistant to possible attacks and allowing them to be used with larger blocks of data, up to 2^{128} bits.

The SHA-2 hash function is composed by the SHA (256/224, SHA/512) hash algorithm, the differences between the versions are in the size of the final digest message the initialization vectors, and the size of the operands, All descriptions of SHA-256, SHA-384 and SHA-512 algorithms can be found in the official NIST standard[18]. Table 1 show a comparative study in terms of function characteristics of four hash algorithms[19–21].

Table 1. Functional characteristics of three hash functions

Hash function	256	384	512
Size of hash value (n)	256	384	512
Complexity of the best attack	2^{128}	2^{192}	2^{256}
Message size	$<2^{64}$	$<2^{128}$	$<2^{128}$
Message block size (m)	512	1024	1024
Word size	32	64	64
Numbers of Words	8	8	8
Digest rounds Number	64	80	80
Constants Kt Number	64	80	80

The SHA-256 and SHA-512 algorithms are very similar. A pseudo code for this algorithm is given in listing 1:

For i =1 to N
{
Initialize registers a, b, c, d, e, f, g, h with the $(i-1)^{st}$ intermediate hash value.

SHA-256	SHA-512
For t= 0 to 63:	For t= 0 to 79:
{	{
$T = h + \Sigma_1^{256}(e) + Ch(e, f, g) + K^{\{256\}}_j + W_j$	$T = h + \Sigma_1^{256}(e) + Ch(e, f, g) + K^{\{256\}}_j + W_j$
$T2 = + \Sigma_0^{256}(a) + MAJ(a, b, c)$	$T2 = + \Sigma_0^{256}(a) + MAJ(a, b, c)$
0	0
h = g	h = g
g = f	g = f
f = e	f = e
e = d + T1	e = d + T1
d = c	d = c
c = b	c = b
b = a	b = a
a = T1 + T2	a = T1 + T2
}	}

3. Countermeasure

The countermeasures are based on some types of redundancies: temporal, information and hardware redundancies. In hardware redundancy, we duplicate a copy of the hardware are implemented to make the same calculate on the same information we duplicate a copy of the hardware implementation. This method offered 100% fault coverage and it can be detect transient fault and permanent fault. Yet, this technique is not appropriate for many applications because it has at least 100% resource overhead.

The method of the time redundancy, we are used the same input information for repeat the same calculation in the same hardware. This method uses minimum hardware overhead, and is appropriate for detecting transient faults and it has a 100% timing overhead.

In the information redundancy the implementation cost varies depending on security level detect all faults. These methods use some supplementary information such as parity bits and other fault detection codes to detect errors in a system. This type of countermeasure allows a trade-off between the security of the SHA and the implementation cost.

In[21], Bahramali et al proposed a novel time-redundancy-based fault diagnostic scheme for the implementation of SHA-1 and SHA-512 round computations.

Abdulhadi et al presented[22] a fault model is based on flipping two bits to decreases the tour number of the SHA-512 function. To detect and prevent the attacks, countermeasures have been proposed. An FPGA implementation used for evaluated the attacks and the countermeasure.

Imtiaz et al proposed in[23] detailed analysis of the propagation of errors to the output in the hardware implementation of SHA-512. This analysis includes the single, transient as well as permanent faults that may appear at any stage of the hash value computation. They proposed an error detection scheme based on parity codes and hardware redundancy.

4. Error Analysis

In this section, we discussed the robustness of the unprotected SHA (256/512) against single and multiple faults. The purpose of the error analysis is to comprise the impact of a fault inserted in the SHA operations. We considered the faults are transient as well as permanent and their effect on the hash process is analyzed. Experiments faults analysis consist to injected one faulty bit in the operation input and determinate the faulty bit number in the output of the same operations. Those faults were injected randomly in each operation.

The addition operation is based on the CSA[2425] architecture. A faulty bit in any one of the input addition results a faulty output. Table 2 presents the effect of fault attacks on the addition operation. This fault attacks consist to introduce one error in one of the 3 inputs of the addition operation.

As presented in Table 2, if a fault is injected in the input of the addition operation, it results an erroneous pseudo-sum and erroneous pseudo-carry in 100% and 50% of the case respectively. This means that the fault masking takes place in addition operation.

The Σ_0 function and the function Σ_1 are based on the rotate operations and a faulty one bit in the Σ input generates only one faulty output bit. The faulty bit position in the output is modified due to rotation process.

Figure 3 presents the effect of one bit injected in the input W_{j-2} of σ_1 operation. The σ_1 operation performs the [SHR 7(x)] function where the message data is shifted 7 bits to the right. This means that the last seven bits in the message data is ignored which explain that the output results is zero error in 2.76% of the fault attack cases.

The CH operation is composed of three input, one of them had be selected or the injection of a fault. The operation $Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$ is such that the fault in the output is undetected due to fault masking caused by the carry output of CSA.

Table 2. Propagation of error in CSA

x	y	z	Correct outputs		Erroneous outputs (error in z)	
			Pseudo-sum $x \oplus y \oplus z$	Pseudo-carry $xy + xz + yz$	Pseudo-sum $x \oplus y \oplus z$	Pseudo-carry $xy + xz + yz$
0	0	0	0	0	1	0
0	0	1	1	0	0	0
0	1	0	1	0	0	1
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	1	0
1	1	0	0	1	1	1
1	1	1	1	1	0	1

4.1 Error Analysis of a Digest Round

The fault attacks were performed against the W_j using single error. The data message length is 1024 bits and 512 bits for SHA-512 and SHA-256 respectively. To simulate the robustness of the digest round, only one bit of the data message injected.

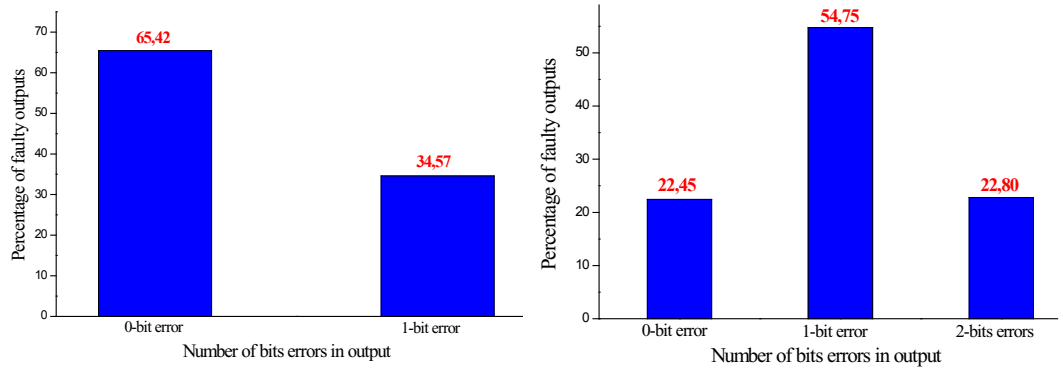
4.2 Error Analysis of a Block Round

The used message consists of 20 data blocks where the output of each blocks is 1024-bit. The number of erroneous bits is 3200-bit.

The CH function consists of three inputs where only one input (input x) is affected with a single errors. The single-bit error attacks effect against the CH function is presented in Figure 1(a). The results show that the error masking taking place in 65% of fault attacks cases. Then we conduct a multiple faults attacks where two input bits of CH have been chosen for the injection of an error. The attacks results are shown in Figure 1(b). The results are zero error output in 22,44%, one error output in 54,75% and two bit error output in 22,8% of fault attacks cases.

The MAJ function is computed as $MAJ(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$. The single-bit error attacks effect against the MAJ function is presented in Figure 2(a). The result show zero error output in 65.89%, one error output in 34.11% of fault attacks cases. The multiple fault attacks results against the MAJ function are shown in Figure 2(b)., and it shown that and two bit error output in 11.56% of fault attacks cases.

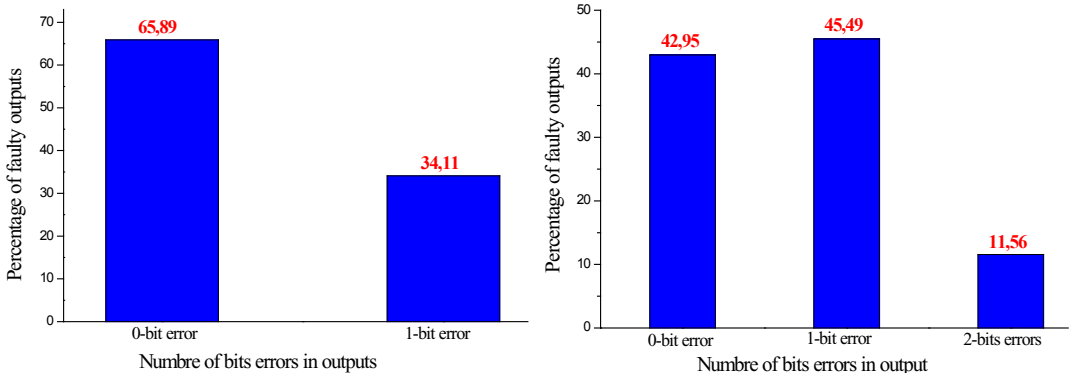
Figure 3 shows the single-bit error attacks result against the σ function. In 72% of fault attacks cases, one bit of σ output was in error.



(a) (b)

Figure 1. (a)ERROR : 1-bit is injected in 'x'. (b) ERROR : 2-bit are injected in 'x' and 'z'.

Figure 1. Distribution of output faults in CH function.



(a) (b)

Figure 2. (a)ERROR : 1-bit is injected in 'x'. (b) ERROR : 2-bit are injected in 'x' and 'z'.

Figure 2. Distribution of output faults in MAJ function.

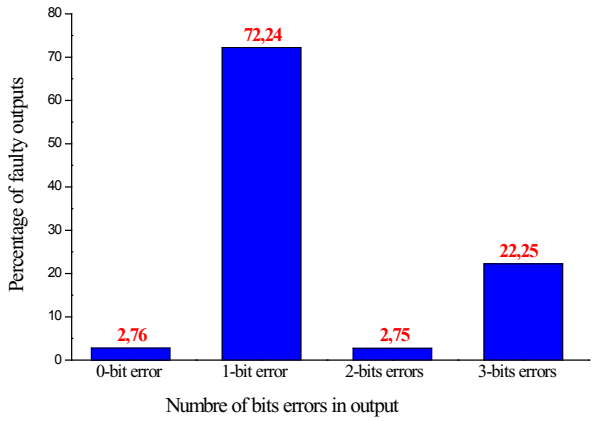


Figure 3. Distribution of output faults in SEGMA function.

5. A Fault Detection Scheme for SHA-256 and SHA-512 Round Computations

The SHA fault detection scheme is implemented to improve the protection of SHA function against the fault attacks. To improve the robustness of SHA implementation, a few fault detection schemes have been proposed. To propose a fault detection scheme that realizes a trade-off between the security of the SHA and their performances, we study some previous proposed methods to secure the SHA. The performances of those methods are studied and compared in terms of detection capabilities percentage and the hardware cost. Until now, the proposed solutions in literature are based on three types; hardware, temporal and information redundancies.

The SHA-256 hash function takes an arbitrary-length input message and generates a 256-bit output. The arbitrary-length input message is divided into a data blocks of 512-bit where each block expansion generates 64 × 32-bit. The SHA-2 round computation comprises of addition, logical rotation and these functions Σ_0, Σ_1, CH and MAJ. The final hash value is obtained by add the last round output to the other blocks outputs. For the SHA-512, it generates a hash output of 512-bit data length where the data blocks length is 1024-bit. The SHA-512 round function consists CH, MAJ, σ_0 and σ_1 functions, the addition operations and the logical rotation. For the SHA-256 is performed consists of 64 round computations where the SHA-512 is realized in 80 round computations.

In this section, in order to protect the SHA-256 and SHA-512 against the transient fault attacks, we proposed time redundancy fault detection scheme. In the SHA-256 and SHA-512, the most important operation is the addition. Figure 4(a) presents a normal time redundancy for this operation.

The C(t1) is the results of adding, at time t1, the two operands A and B while the C(t2) is the results of adding, at time t2, the two operands A and B. Then, the addition compares the two outputs (C(t1) and C(t2)) to detect only the transient faults attacks.

To contain this problem, we use the method shown in Figure 4(b) where A and B are added at time t1, i.e., C(t1) = A(t1) + B(t1). Then at t2 the input B(t2) is subtracted from C(t2). Then, a comparison is performed between A(t1) to detect all the faults in the adder.

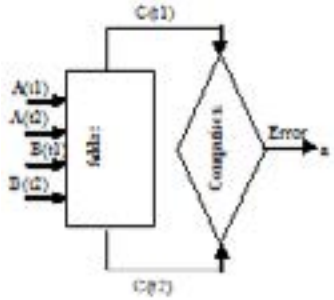


Figure 4(a). Normal time redundancy.

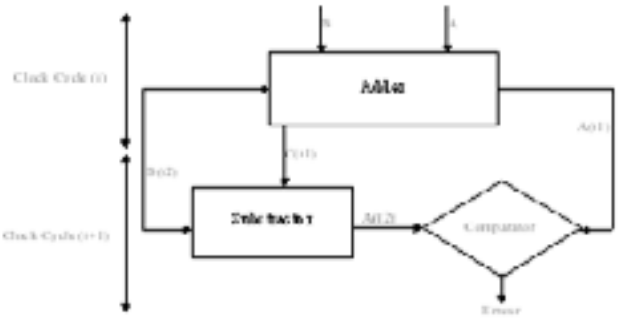


Figure 4(b). The proposed time redundancy.

Figure 4. Time redundancy in an adder.

5.1 Protection of Σ System

This work present a fault detection scheme to protect the Σ functions against the fault attacks which based on the (n+1,n) Cyclic Redundancy Check (CRC). The variable n is the number of byte in the encryption data. All detection operations are performed over GF(2^8), while the generator polynomial of degree (n-1) is:

$$g(x) = x + 1 \quad (1)$$

The SHA implementation consists of two sum function Σ_0 and Σ_1 which based rotates operations. If a faulty bit

is injected in the Σ function, a single erroneous bit will be produced in the output. The erroneous bit position is determinate due to rotate operation. The sum Σ_0 consists of three rotations: ROR^{28}, ROR^{34} and ROR^{39}. If an erroneous bit is injected in the Σ_0 function, three erroneous bits will be produced in the output. In the functions Σ_0, the rotations transformation can be expressed as:

$$ROR_i^{28} = ROR_0^{28} \oplus ROR_1^{28} \oplus ROR_2^{28} \oplus ... \oplus ROR_7^{28} \quad (2)$$
$$ROR_i^{34} = ROR_0^{34} \oplus ROR_1^{34} \oplus ROR_2^{34} \oplus ... \oplus ROR_7^{34}$$
$$ROR_i^{39} = ROR_0^{39} \oplus ROR_1^{39} \oplus ROR_2^{39} \oplus ... \oplus ROR_2^{39}$$

Once we obtain different rotation equations, we calculate Σ_0 from the equation below:

$$\sum_{i=0}^{3} \Sigma_0 = \sum_{i=0}^{3} ROR^{28}(i) \oplus \sum_{i=0}^{3} ROR^{34}(i) \oplus \sum_{i=0}^{3} ROR^{39}(i) \quad (3)$$

Figure 5 has shown our proposed detected scheme for Σ_0.

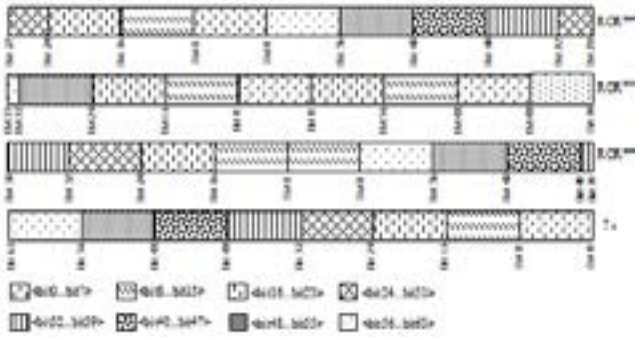


Figure 5. Proposed solution for sum Σ_0.

6. Error Detection

To evaluate of the proposed scheme for the hash implementation, we have considered two types of faults: single faults and multiple faults.

6.1 Single Fault Attacks

In the first fault attacks, we considered the single faults where only one byte of the message data is affected. For this propose, we proposed the simulation model described in Figure 6.

The generated faults using the proposed model are different in terms of erroneous bits in one data byte. Those faults have been introduced in all possible location: every byte, every transformation and at every round.

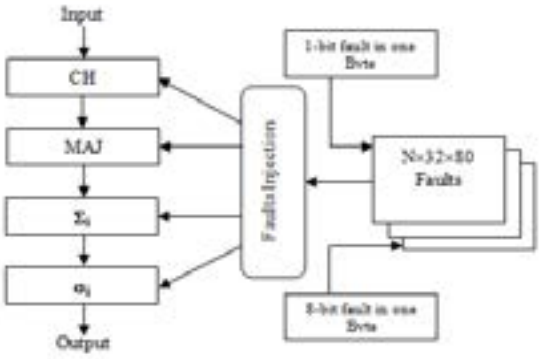


Figure 6. Simulation model for single faults.

As presented in Table 3, our detection scheme, for the hash implementation, can detect all even-bit and odd-bit faults. The undetectable faults percentage reaches 0% against single faults attacks. This means that the detection capability percentage achieves 100%. Thus, we conclude that our scheme for the SHA ensures sufficient security against the single fault attacks.

Table 3. Detection capability of the proposed fault detection scheme: single fault attacks

Fault multiplicity	Number of injected faults	Number and percentage of the detected faults
1	5120	100%
2	17920	100%
3	35840	100%
4	44800	100%
5	35840	100%
6	17920	100%
7	5120	100%
8	640	100%
Average	-	100%

6.2 Multiple Fault Attacks

In the second experiments, we considered the multiple faults where more than 2 bits in 2 different bytes of the message data is affected. For this propose, we proposed the simulation model described in Figure 7. The number of erroneous bits varied from 1 to 20.

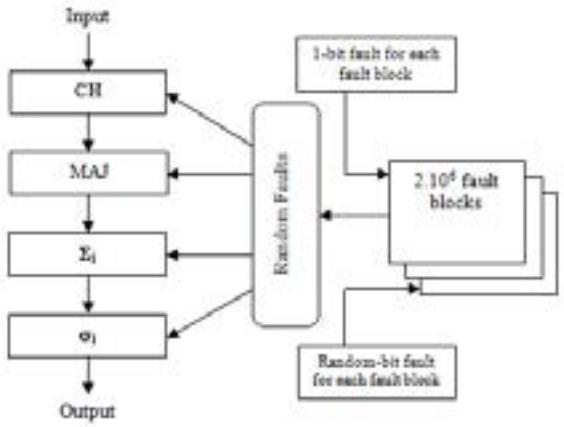


Figure 7. Simulation model for multiple faults.

The proposed solution for SHA is simulated by 21 tests different in terms of erroneous bits in the 512-bit message data. The protected solution is simulated using 2.000.000 random faults in every test. These faults have been injected in all possible location: every byte, every transformation and at every round. The multiple fault attacks results are presented in Figure 8.

As seen in the Figure 8, all injected faults with odd erroneous bit were detected. The detection capability percentage is always 100%. Figure 8 show also that the undetectable faults percentage decreased dramatically when erroneous bit number increased.

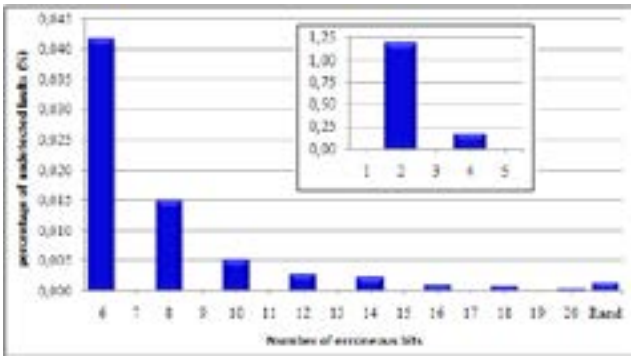


Figure 8. Percentage of the undetected multiple faults injected in the hash implementation.

As for as, the percentage of undetected errors up to 1,19% when the multiplicity is 2 when the multiplicity is 8, the percentage up to 0,0150%. For the last test named random, the percentage is 0,001%. Thus, we conclude that our scheme for the SHA ensures sufficient security against multiple fault attacks.

7. Implementation of the Proposed Fault Detection Scheme

In this section, we present the FPGA implementation results of the protected SHA and a comparison with the unprotected SHA. The two implementations have been modeled using VHDL, simulated and synthesized using by ModelSim 6.6 and Xilinx ISE 10.1.03 respectively. The FPGA platform used is the Xilinx Virtex-5. Table 4 presents the implementation results.

As seen in Table 4, the unprotected SHA requires 922 slices for 151,88 MHz maximal Frequency while the protected SHA necessitates 11,93 % more slices than the unprotected SHA. The FPGA implementation results show, also, that our fault detection scheme does not affect the working frequency of the SHA function which means the proposed detection scheme ensures high implementation performances.

Table 5 presents comparison between the proposed fault detection schemes with other work. Since, the most of the previous works are presented an FPGA implementation on VIRTEXII Pro, we implemented the proposed fault detection scheme on this device. FPGA implementation resultants show our proposed method occupies 33,52% less slices and the frequency augments about 45% than the work presented in[21]. Table 5 show, also that our proposed design results about 314, 39% throughput overhead than[21]. At security level, the proposed detection scheme achieves 99,999% detection capability while the scheme in[21] achieves 99% fault coverage. Therefore, our

Table 4. FPGA implementation of the proposed fault detection scheme: results

SHA-512	Area (Slice) (overhead)	Frequency (MHz) (degradation)	Throughput (Mbps) (degradation)	Efficiency (Mbps/slice)
Without fault detection	922	151,883	1829,743	1,985
With fault detection	1032 (11,931%)	151,883 (»0%)	1829,743 (»0%)	1,773

Table 5. FPGA implementation of the proposed fault detection scheme: results and comparison

SHA-512	Fault coverage %	Area (Slice) (degradation)	Frequency (MHz) (overhead)	Through (Mbps) (overhead)	Efficient (Mbps/slice) (overhead)
[21]	99%	2062	76	477	0,23
Proposed	99,999%	2692 (33,52%)	117,155 (45,15%)	1499,584 (314,39%)	0,55 (139,13%)

fault detection scheme has the highest performances in terms of fault coverage, throughput, frequency and area.

8. Conclusion

This work presents a new SHA countermeasure against faults attacks based on the hybrid redundancy. We analyze the weaknesses and the strengths of this countermeasure against the single and multiple fault attacks. The SHA countermeasure performances in terms of frequency degradation and area overhead for the module have been extracted and compared. The fault attacks results show that our SHA countermeasure has 100% fault coverage against the single fault attacks and 99,999% fault coverage against the random faults. FPGA implementation results present that the frequency and throughput degradation for the hash process are 45,15% and 314,39% respectively. Hence, our fault detection scheme has the highest performances in terms of fault coverage, throughput, frequency and area.

9. References

1. National Institute of Standards and Technology. Secure Hash Standard (SHS). Federal Information Processing Standards Publication (FIPS PUB). 2008 Oct; 180(3):3–27.
2. Ahmad I, Das AS. Analysis and detection of errors in implementation of SHA-512 algorithms on FPGAs. The Computer Journal. 2007 Jun 2; 50(6):728–38. Crossref
3. Bar–El H, Choukri H, Naccache D, Tunstall M, Whelan C. The sorcerer's apprentice guide to fault attacks. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE). 2006 Feb; 94(2):370–82. Crossref.
4. Bertoni G, Breveglieri L, Koren I, Maistri P. An efficient hardware-based fault diagnosis scheme for AES: performances and cost. In the Proceedings of the 19th Institute of Electrical and Electronics Engineers (IEEE) International Symposium on Defect and Fault Tolerance in VLSI Systems, France; 2004 Oct 10–13. p. 130–8. Crossref.
5. Chaves R, Kuzmanov G, Sousa L, Vassiliadis S. Cost efficient SHA hardware accelerators. Institute of Electrical and Electronics Engineers (IEEE) Transactions on Very Large Scale Integration (VLSI) Systems. 2008 Aug; 16(8):999–1008. Crossref
6. Crowe F, Daly A, Kerins T, Marnane W. Single-chip FPGA implementation of a cryptographic co-processor. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) International Conference on Field-Programmable Technology; 2004 Dec 6–8. p. 279–85. Crossref.
7. National Institute of Standards and Technology. Cryptographic hash algorithm and SHA-3 competetion [Internet]. 2010 [cited 2010 Dec 13]. Available from: Crossref.
8. Dadda L, Macchetti M, Owen J. The design of a high speed ASIC unit for the hash function SHA-256 (384, 512). In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) Design, Automation and Test in Europe Conference and Exhibition. 2004 Feb 16–20; 3:70–5. Crossref
9. Grembowski T, Lien R, Gaj K, Nguyen N, Bellows P, Flidr J, Lehman T, Schott B. Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA- 512. In the 5th International Conference on Information Security (ISC); 2002 Sep 5. p. 75–89.
10. Juliato M, Gebotys C, Elbaz R. Efficient fault tolerant SHA-2 hash functions for space applications. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) Aerospace Conference, USA; 2009 Mar 7–14. p. 1–16. Crossref.
11. Samyde D, Quisquater J. Eddy current for magnetic analysis with active sensor smart card programming and security. In the Proceedings of the Esmart; 2002. p. 185–94.
12. Skorobogatov SP, Anderson RJ. Optical fault induction attacks. International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Lecture Notes in Computer Science, Springer. 2002; 2523:2–12.
13. Kommerling O, Kuhn MG. Design principles for tamper-resistant smartcard processors. USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA; 1999 May 10–11. p. 1–13.
14. Anderson R, Kuhn M. Low cost attacks on tamper resistant devices. International Workshop on Security Protocols (IWSP), Lecture Notes in Computer Science, Springer. 1997; 1361:125–36.
15. Peterson I. Chinks in digital armor: exploiting faults to break smart-card cryptosystems. Science News. 1997 Feb 1; 151(5):78–9.
16. National Institute of Standards and Technology. Secure hash standard. Federal Information Processing Standards Publication. 2002 Aug 1; 180(2):3–71.
17. Wang X, Yin YL, Yu H. Finding collisions in the full SHA-1. Annual International Cryptology Conference (CRYPTO), Advances in Cryptology, Lecture Notes in Computer Science, Springer. 2005; 3621:17–36.

18. Canniere CD, Rechberger C. Findin SHA1 Characteristics: General Results and Applications. In the 12th International Conference on the Theory and Application of Cryptology and Information Securit (ASIACRYPT), Lecture Notes in Computer Science, Springer. 2006; 4284:1–20.
19. Dadda L, Macchetti M, Owen J. The design of a high speed ASIC unit for the hash function SHA-256 (384, 512). In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) Design, Automation and Test in Europe Conference and Exhibition. 2004 Feb 16–20; 3:70–5.
20. National Institute of Standards and Technology. Secure hash standard. Federal Information Processing Standards Publication. 1995 Apr 17; 180(1):1–10.
21. Bahramali M, Jiang J, Reyhani-Masoleh A. A fault detection scheme for the FPGA implementation of SHA-1 and SHA-512 round computations. Journal of Electronic Testing, Springer. 2011 Aug 2; 27:517.
22. Abdulhadi S. A fault attack on a hardware-based implementation of the secure hash algorithm SHA-512. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) International Conference on Reconfigurable Computing and FPGAs (ReConFig), Cancun, Mexico; 2013 Dec 9–11. p. 1–7.
23. Ahmad I, Das AS. Analysis and detection of errors in implementation of SHA-512 algorithms on FPGAs. The Computer Journal. 2007 Jun 2; 50(6):728–38. Crossref.
24. Kahri F, Mestiri H, Bouallegue B, Machhout M. Enhanced FPGA implementation of the SHA-512 hash function. International Review on Computers and Software (IRECOS). 2014 Nov; 9(11): p.1816–21
25. Siewiorek DP, Swarz RS. Reliable computer systems (3rd edition): design and evaluation. A. K. Peters Ltd, Natick, USA; 1998 Dec 15. p. 908.