Enhancing IoT Security for Sustainable Development: A Parity Checking Approach for Fault Detection in PRESENT Block Cipher

Nada Maatallah
Electronics and Micro-Electronics Laboratory, Faculty of Sciences of Monastir, University of Monastir, Tunisia
nadamaatallah00@gmail.com

Hassen Mestiri
Department of Computer Engineering, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
h.mestiri@psau.edu.sa (corresponding author)

Abdullah Alsir Mohamed
Department of Computer Engineering, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
a.mhamed@psau.edu.sa

Mohsen Machhout
Electronics and Micro-Electronics Laboratory, Faculty of Sciences of Monastir, University of Monastir, Tunisia
mohsen.machhout@fsm.rnu.tn

Received: 1 January 2025 | Revised: 28 January 2025 and 17 February 2025 | Accepted: 27 February 2025

 DOI: https://doi.org/10.48084/etasr.10109

ABSTRACT

The PRESENT lightweight block cipher designed for resource-constrained environments exhibits vulnerabilities to fault injection attacks. By deliberately introducing errors during the computation, attackers can potentially recover secret keys or bypass security measures. Various fault models, including single- and multi-bit faults targeting different stages of the cipher, have been explored, demonstrating the feasibility of such attacks. Consequently, robust countermeasures, such as error detection codes, parity checks, and hardware redundancy, are essential to enhance the fault resistance of PRESENT implementations and maintain security in real-world deployments. This paper presents an enhanced fault detection scheme for the PRESENT lightweight block cipher, designed to provide a high level of protection against a wide range of fault injection attacks. The proposed scheme focuses on detecting both simple and multiple fault attacks, addressing scenarios that target one or more bytes. A comprehensive analysis of the detection capabilities is performed, considering various fault multiplicities and injection methods. This innovative approach contributes to the advancement of secure and reliable systems, in line with the focus of SGD 9 on fostering innovation. The proposed scheme is extensively evaluated through simulations, demonstrating its ability to detect a significant percentage of injected faults. A hardware implementation on a Xilinx Virtex5-XC5VFX70T FPGA platform is explored, analyzing the trade-off between security, area, and performance. The results show that the proposed scheme achieves high fault coverage while maintaining reasonable resource utilization without impacting operating frequency. A comparison with existing techniques highlights the advantages of the proposed approach.

Keywords-security; cryptography; PRESENT block cipher; lightweight algorithm; fault attacks; encryption algorithm; secure communication

I. INTRODUCTION

PRESENT lightweight cryptographic algorithms are essential for resource-constrained IoT devices, designed to provide secure communication and data protection without draining their limited power supply [1]. Its efficiency and simplicity make it ideal for a wide range of applications in the rapidly expanding IoT ecosystem. Lightweight cryptographic algorithms are also suitable for use in medical devices, ensuring the confidentiality and integrity of sensitive patient data. Furthermore, these algorithms can be utilized in industrial IoT devices to secure communication and prevent unauthorized access to critical infrastructure. Deploying the PRESENT cipher in a hardware implementation can increase its susceptibility to fault injection attacks [2-3]. These attacks exploit vulnerabilities in hardware implementations to induce faults and compromise security [4-8]. Therefore, robust countermeasures are crucial to ensure the reliability and security of these devices. Recent research has focused on developing efficient countermeasures against fault injection attacks on the PRESENT lightweight algorithm [9-11]. These studies have proposed various fault detection schemes, such as information redundancy, hardware redundancy, and temporal redundancy, to safeguard the PRESENT algorithm.

In [9], a new design strategy was proposed for lightweight cryptographic primitives resistant to both side-channel and fault attacks, particularly relevant to resource-constrained IoT devices. This study focused on techniques such as fault space transformation, which uses redundancy and diffusion layers to make fault injection more challenging. Additionally, it highlighted the importance of designing security from the ground up, especially in resource-limited environments, and used AES-128 as a case study to illustrate the effectiveness of FST in increasing the Hamming distance between equivalent faults in original and redundant computations. The results of the experimental hardware implementation indicated that the unprotected PRESENT implementation operated at a frequency of 205 MHz, while the protected PRESENT version reached a frequency of 65.7 MHz. This corresponded to a frequency reduction of 67.9%.

In [10], the effectiveness of concurrent error detection codes was investigated against Laser Fault Injection (LFI) attacks on FPGA-based cryptographic implementations. An LFI setup was used to target a CED-protected block cipher implemented on a Xilinx Virtex-5 FPGA. The results showed a high probability (99.293%) of inducing an even number of bit flips with a single laser pulse, rendering single-bit parity checks ineffective. Although randomized parity schemes offered better protection, non-linear codes did not significantly outperform linear codes, despite higher overhead. This study highlighted the limitations of standard CED against LFI and suggested that linear randomized parity codes offer a good balance between security and cost.

In [11], a PLL/RO-based detection system was explored for LFI attacks on FPGAs. The ring oscillator's frequency, monitored by a phase-locked loop, was disrupted by laser attacks, triggering an alarm. Experimental results on a Xilinx Virtex-5 FPGA demonstrated a high detection rate of up to 92.82% when protecting registers within a slice. Additionally, the countermeasure could detect attacks before they caused faulty cipher execution. Specifically, the lowest laser power to trigger the alarm was 64% of the laser's maximum power, while fault injection in data registers required at least 75% (with outliers) and mostly above 98%. This indicated a good security margin. Further analysis showed a detection rate of approximately 99.06%, meaning that only 0.94% of injections went undetected by the countermeasure and still affected the cipher.

In [11], a low-area implementation of the PRESENT cipher was presented on an FPGA, enhancing security by incorporating a true random and a pseudo-random number generator for key generation. This approach generated unique keys for each round of encryption, increasing resistance against attacks. The design utilized a dual port ROM to optimize hardware resource utilization and achieve a higher operating frequency. The performance of this implementation was evaluated using metrics such as slice registers, flip-flops, LUTs, and power consumption. The quality of the recovered input image was also assessed using PSNR, SSIM, and MSE.

In [13], various hardware architectures for the PRESENT block cipher were explored along with their respective FPGA implementations. Different design strategies were investigated, including iterative and pipelined architectures, with varying datapath widths, to optimize throughput and area efficiency. The trade-offs between these architectures were analyzed, considering factors such as latency, power consumption, and resource utilization on the FPGA. A comprehensive performance evaluation of these implementations was performed, offering insights into the suitability of different architectures for various resource-constrained environments.

In [14], two novel parity-check-based methods were proposed for detecting timing fault injection attacks: mixed-grained parity check and word recombination parity check. These methods aimed to balance security and overhead, addressing the limitations of traditional parity checks. The mixed-grained parity check applied fine-grained checks to security-critical operations and coarser checks elsewhere, while a word recombination parity check reorganized subwords before checking, effectively simulating fine-grained checks with lower overhead. These methods were evaluated on RC5, AES, and DES implementations, demonstrating improved fault coverage compared to traditional parity checks with manageable resource costs.

This study presents an enhanced fault detection scheme for the PRESENT lightweight block cipher, based on the use of parity checking as a means of detecting faults. The proposed scheme focuses on detecting both simple and multiple fault attacks, addressing scenarios that target one or more bytes. This study contributes to the resilience and security of the PRESENT lightweight algorithm, aligning with the goals of SDG 9 to build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation. A comprehensive analysis of the detection capabilities considered various fault multiplicities and injection methods.

II. BACKGROUND

A. PRESENT Lightweight Block Cipher

The PRESENT block cipher is a symmetric key algorithm that encrypts data in fixed-size blocks. It was designed by researchers at Orange Labs and is widely used in applications such as SSL/TLS for secure communication over the Internet [1]. PRESENT operates on a 64-bit block size and supports key lengths of 80 or 128 bits. Its lightweight design makes it well-suited for resource-constrained devices such as RFID tags and sensor networks. PRESENT is known for its efficient performance and low memory requirements, making it ideal for applications with limited processing power and memory. Additionally, its resistance to various cryptanalytic attacks ensures the security of encrypted data. The PRESENT consists of a substitution-permutation network with 31 rounds, providing a high level of security against attacks. Its simplicity and efficiency make it a popular choice for embedded systems and IoT devices. The PRESENT round functions are: sBoxlayer, permutation layer (pLayer), and key addition layer (addRoundKey). These components work together to provide strong encryption while minimizing resource usage, making PRESENT a practical solution for devices with limited capabilities. The substitution layer (sBoxlayer) is responsible for the nonlinear mixing of the input data, increasing the complexity of the encryption process. The pLayer then shuffles the data to add another layer of confusion, making it even more difficult for attackers to decipher the encrypted information. Additionally, the addRoundKey layer introduces a unique key at each round to further enhance security and prevent unauthorized access to the encrypted data. Overall, the combination of these three components in the PRESENT round function creates a robust encryption scheme that is well-suited for securing IoT devices and systems.

```
Algorithm 1: The PRESENT Algorithm [1]
generateRoundKeys()
for i=1 to 31 do
  AddRoundKey(STATE,Ki)
  sBoxLayer(STATE)
  pLayer(STATE)
end for
AddRoundKey(STATE,K32)
```

B. Fault Attacks

Fault injection attacks, a critical threat to cryptographic systems, involve the deliberate introduction of errors during computation to manipulate the system's behavior and potentially expose sensitive information. These attacks utilize various methods, including clock and power glitching, electromagnetic and laser fault injection, and temperature variations, to disrupt the intended flow of operations [15]. By analyzing the resulting erroneous outputs, attackers can deduce secret keys, bypass security measures, cause denial-of-service conditions, or corrupt data. Protecting against these attacks requires a multilayered approach encompassing error detection codes, hardware redundancy, physical shielding, and specialized detection circuits, both in hardware and software [16].

III. PROPOSED FAULT DETECTION SCHEME

A detection system is proposed to protect the execution of the sBoxLayer, which exploits the relationship between the input and output of this transformation set. For both the pLayer and addRoundKey, parity is used as an error detector during the operation process of these transformations. The data parity check is carried out at a low cost since the parity check is adopted for these transformations. Let M be the number of bytes in a PRESENT data message. The proposed error detection system can be applied using two methods.

- First method ($M = 4$): The error detection system is applied to half of the 64-bit block, specifically 32 bits. The data block is divided into two parts, and the system checks one part for errors at each step. This approach helps reduce complexity and accelerate detection by analyzing a smaller subset of data, although it provides only partial coverage at each stage.
- Second method ($M = 8$): The detection system is applied to the entire 64-bit block (8 bytes). This means that all the data within the block is checked only once to detect errors. This method provides complete coverage for each verification system. Figure 1 presents the general structure of the proposed fault detection scheme. The detailed implementation of each present transformation follows.

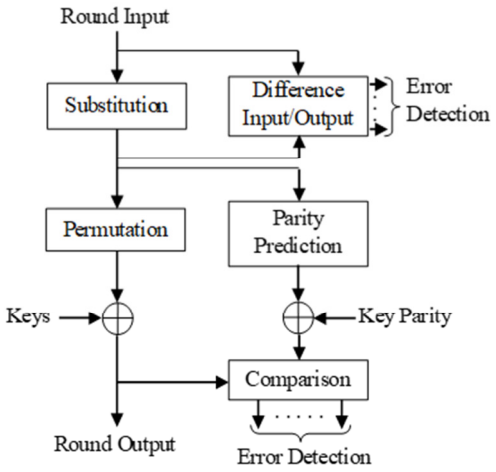

Fig. 1. PRESENT fault detection scheme principle.

A. Non-Linear Substitution Layer (sBoxLayer) Protection

The sBoxLayer transformation, which is the only nonlinear transformation in the PRESENT_80 algorithm, consists of 16 S-Boxes. These S-Boxes can be implemented either using lookup tables or combinatorial logic. The use of parity as an error detection code is straightforward due to the nonlinearity of the system transformation, which involves a substitution operation on each 4-bit block.

Let D be the difference between the input β and the output β' of the sublayer transformation. Since the value of a 4-bit block can vary between 0 and 15, the 16 possible values of D are pre-calculated and stored in a table. The difference D is calculated as follows:

$$D_{i,j} = \beta_{i,j} \oplus \beta'_{i,j} \quad (1)$$

with $0 \leq i, j \leq 3, D_{i,j}, \beta_{i,j}, \beta'_{i,j} \in GF(2^4)$.

The errors are detected by the error detection flags. These flags are obtained by comparing the difference D with the value $\beta_{i,j} \oplus \beta'_{i,j}$. The 16 error detection flags ($E_{i,j}$) of the sublayer transformation can be calculated as follows:

$$E_{i,j} = D_{i,j} \oplus \beta_{i,j} \oplus \beta'_{i,j} \quad (2)$$

with $0 \leq i, j \leq 3$, and $E_{i,j} \in GF(2^4)$.

As shown in Figure 2, 16 error detection flags are generated during the execution of the sublayer transformation (an error detection flag for each S-Box). If one of the 16 error detection flags is set to 1, it means that at least one fault is detected during the execution of the sublayer transformation.

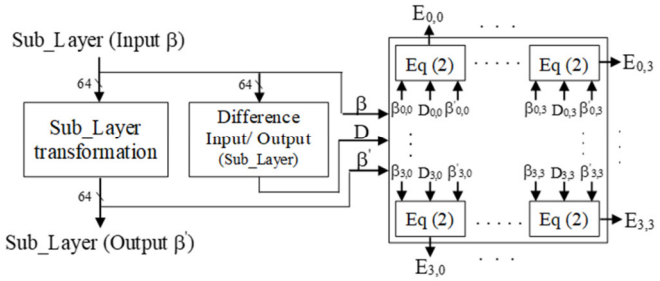

Fig. 2. Fault detection scheme for sBoxLayer.

The proposed detection system is independent of the way the sublayer is implemented (LUT or combinatorial logic).

B. Permutation Layer (pLayer) Protection

The output of the pLayer transformation acts as input for the permutation transformation. The latter permutes the bits of the state, but it does not affect their values. In the case of M = 8, the parity of the input P_{PEin} and the parity of the output P_{PEout} of the permutation transformation are calculated as follows:

$$P_{PEin} = \sum_{i=0}^{63} P_i$$

$$P_{PEout} = \sum_{j=0}^{15} \sum_{i=0}^{3} P_{((16\times i)+j)} \quad (3)$$

The faults are detected by comparing P_{PEin} to P_{PEout}, and any differences found are reported as faults. When M=4, the fault detection scheme checks for faults in 32-bit data and P_{PEin} and P_{PEout} are calculated as follows:

$$P_{PEin} = \sum_{l=0}^{1} \sum_{i=32\times l}^{32\times l+31} P_i$$

$$P_{PEout} = \sum_{l=0}^{1} \sum_{j=8\times l}^{8\times l+7} \sum_{i=0}^{3} P_{((16\times i)+j)} \quad (4)$$

C. AddRoundKey Protection

The AddRoundKey transformation calculates the modulo 2 addition between the state matrix and the key round $K = \{k_i;\ 0 \leq i \leq 63\}$ to obtain the output of the PRESENT round. Consequently, the parity of the output round P_{out} can be easily predicted for M = 8 according to the following equations:

$$P_K = \sum_{i=0}^{63} K_i$$

$$P_{PE} = \sum_{i=0}^{63} K_i \oplus \sum_{j=0}^{15} \sum_{i=0}^{3} P_{((16\times i)+j)} =$$

$$= P_K \oplus P_{PEout} \quad (5)$$

where P_k is the addRoundKey parity and P_E is the PRESENT round output parity.

The faults are detected by comparing P_{PE} to $P_K \oplus P_{PEout}$, and any differences found are reported as faults. When M = 4, the fault detection scheme checks for faults in 32-bit data and P_K and P_{PE} are calculated as follows:

$$P_K = \sum_{l=0}^{1} \sum_{i=32\times l}^{32\times l+31} K_i$$

$$P_{PE} = \sum_{l=0}^{1} \sum_{i=32\times l}^{32\times l+31} K_i \oplus \sum_{l=0}^{1} \sum_{j=8\times l}^{8\times l+7} \sum_{i=0}^{3} P_{((16\times i)+j)} =$$

$$= P_K \oplus P_{PEout} \quad (6)$$

IV. EVALUATION OF THE ERROR DETECTION CAPABILITY

This section describes the simulation results to evaluate the resistance of the proposed system in the encryption process against the fault injection attack. The injected faults are of transient types and last for one clock cycle.

A. Simple Fault Attacks

First, the error detection capability of different systems was analyzed while ensuring that the attacker targets only one byte. The injected faults are of the transient type and last for one clock cycle. Figure 3 shows the simulation model. Each system is simulated by eight tests distinguished by the multiplicity of the faults. Each error model consists of N blocks, where N is the number of possible errors for each fault multiplicity. The faults were exhaustively injected into each byte, each operation, and at each round. Consequently, each error model is used 8×94 times, where 8 is the number of state matrix bytes and 94 is the number of all operations in the PRESENT rounds. Table I presents the results of the detection capability simulation using ModelSim 6.6.

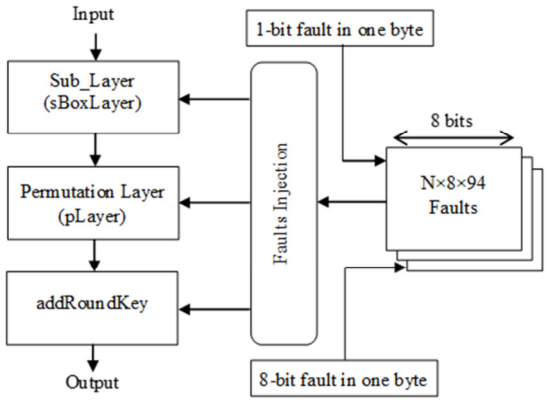

Fig. 3. Simple fault attack simulation model.

Table I shows the fault multiplicity, the number of errors injected for each fault multiplicity, and the number and percentage of undetected errors for each error detection system.

The number of errors injected for each fault multiplicity is calculated as follows:

$$N \times 8 \times 94 \quad (7)$$

where N is the number of possible faults for each fault multiplicity. Faults were exhaustively injected in every byte, every operation, and every round. Therefore, each fault was used 8×94 times, where 8 is the number of bytes of the state and 94 is the number of all operations in all rounds of the PRESENT.

TABLE I. SIMPLE FAULT ATTACK DETECTION CAPACITY

Fault multiplicity	Number of injected faults	Percentage of undetected faults
1	6016	0 (0%)
2	21056	0 (0%)
3	42112	0 (0%)
4	52640	0 (0%)
5	42442	0 (0%)
6	21056	0 (0%)
7	6016	0 (0%)
8	752	0 (0%)
9	-	0 (0%)

As shown in Table I, all even and odd multiplicity errors on the 64-bit data are detected. The percentage of undetected errors reaches 0%, which means that the probability of detecting faults affecting a single byte is 100%. Since most fault injection attacks exploit faults affecting a single byte, the proposed system ensures a high level of security against these types of attacks.

B. Multiple Fault Attacks

In the second step, the detection capabilities of the error detection systems were analyzed while ensuring that the attacker targets more than one byte. This section evaluates the resistance of the proposed system by assuming that the attacker targets more than one octet at a time.

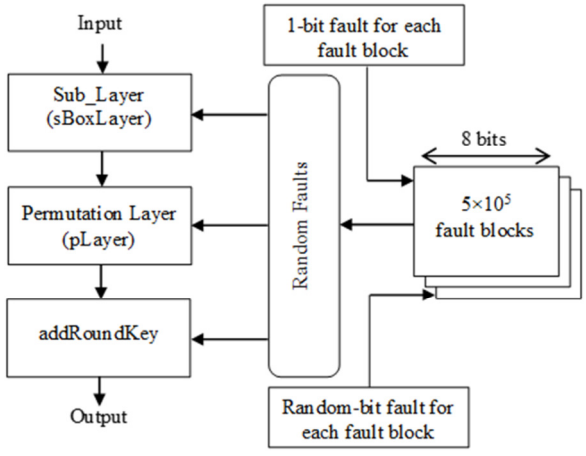

Fig. 4. Multiple fault attacks simulation model.

The considered faults are of transient types, lasting for one clock cycle, with an error multiplicity ranging from 1 to 20 bits. The simulation model described above and shown in Figure 4 was used. The proposed system was simulated by 21 tests distinguished by the number of erroneous bits. During each trial, 500,000 faults were injected to assess the resistance of the proposed system. The affected bits were randomly chosen among the 64 bits of the state matrix. Table II presents the simulation results.

TABLE II. MULTIPLE FAULT ATTACKS DETECTION CAPACITY

Fault multiplicity	Undetected faults percentage (%)	
	$M = 4$	$M = 8$
2	4.7288	10.9991
3	0	0
4	0.6366	3.3761
5	0	0
6	0.1454	1.5724
7	0	0
8	0.0436	0.9798
9	0	0
10	0.0182	0.6828
11	0	0
12	0.0100	0.5370
13	0	0
14	0.0062	0.4562
15	0	0
16	0.0030	0.4174
17	0	0
18	0.0024	0.4060
19	0	0
20	0.0018	0.3830
Random	0.2945	0.5213

As shown in the $M = 4$ column, all errors of odd multiplicity are detected. The percentage of undetected errors decreases inversely proportional to the number of erroneous bits. For the random fault test, the percentage of undetectable errors is approximately 0.2945%. The same data models and test conditions used in the previous tests were used to verify the proposed detection system. In the case of $M = 8$, all errors of odd order were detected, thus the percentage of undetectable errors decreased inversely proportional to the number of erroneous bits. In the Random faults case, the percentage of undetectable errors was approximately 0.5213%. During the execution of the permutation and addRoundKey functions, if the proposed system is applied to each 32-bit PRESENT data ($M = 4$), two error detection flags are generated to secure the encryption process. On the other hand, the protection of the entire 64-bit PRESENT data state data ($M = 8$) generates a single error detection flag. In this case, the faults injected into 64 bits of the data are not detected if their modulo 2 addition is equal to zero. However, in the case of M = 4, even if the modulo 2 addition of these faults is equal to zero and the faults affect at least two 32-bit columns, at least one of the two error detection flags is set to one. Figure 5 presents a comparison between the two proposed error detection scheme versions ($M = 4$ and $M = 8$), in terms of the percentage of undetected errors. Both systems were simulated with the same model and the same test conditions. Given that the values of the two tests were high, they are represented separately in a complementary graph. Thus, both systems detected all odd multiplicity errors. For the random fault test, the proposed system with $M = 4$ achieved a lower rate of undetected faults.

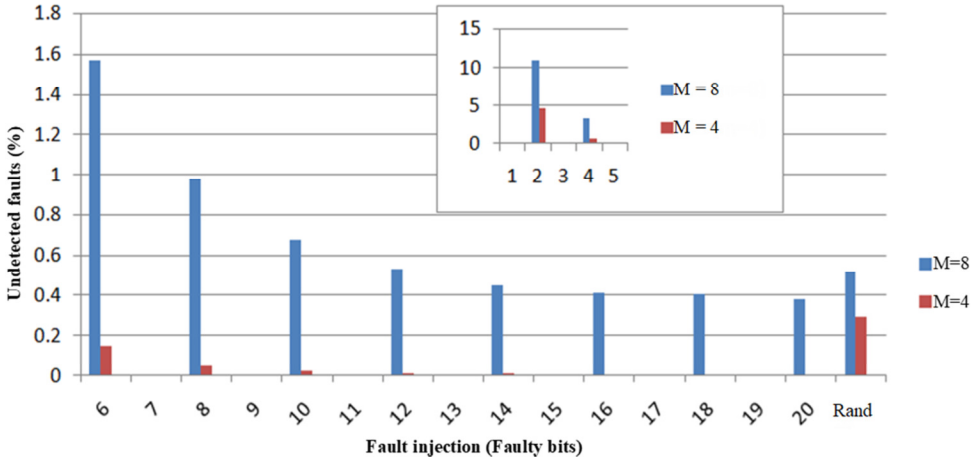

Fig. 5. Percentage of undetected multiple faults injected in the PRESENT module.

V. HARDWARE IMPLEMENTATION OF THE PROPOSED FAULT DETECTION SCHEME

The proposed error detection system was applied to the PRESENT algorithm. The implementation of PRESENT with and without the proposed error detection system was modeled using VHDL, simulated with ModelSim 6.6, and synthesized with Xilinx ISE 14.4. The FPGA platform used was the Xilinx Virtex5-XC5VFX70T. Table III presents the results of the hardware implementation.

TABLE III. PROPOSED FAULT DETECTION SCHEME: RESULTS AND COMPARISON

Fault detection scheme	**FC(%)**	**Area (Slices LUTS) (*Overhead*)**	**Area (Occupied slices) (*Overhead*)**	**Frequency (MHz) (*Degradation*)**	**Throughput (Mbps) (*Degradation*)**
Original	-	345	134	381.018	908.42
M = 8	99.4787	380 *(10.145%)*	141 *(5.224%)*	381.018 *(0%)*	908.42 (0%)
M = 4	99.8527	403 (16.812%)	163 *(21.642%)*	381.018 (0%)	908.42 (0%)

As shown in Table III, the error detection capability (FC), area, frequency, throughput, area overhead, and frequency degradation for the original and secure PRESENT were determined. The implementation of PRESENT without an error detection system occupies an area of 345 LUT slices and an area of 134 occupied slices at a frequency of 381.018 MHz. The protection of this implementation against fault attacks with the error detection system M= 4 results in an area overhead of approximately 16.812% and an increase of 21.642% in the occupied area. Although the proposed system generates an area overhead, the degradation in operating frequency remains constant. However, it provides an error detection capability of approximately 99.8527%.

Thus, the proposed detection system achieves an effective trade-off between security level and hardware implementation performance, while ensuring a high level of security against fault injection attacks without affecting the operating frequency. Table IV compares the proposed fault detection schemes with previously published works.

TABLE IV. FAULT DETECTION SCHEME: COMPARISON (DECREASE IS DENOTED BY USING '-' SIGN)

Fault detection scheme	**Performance overhead (%)**					
	FC(%)	**Area (Slices LUTS)**	**Area (Occupied slices)**	**Frequency (MHz)**	**Throug. (Mbps)**	**Power (mW)**
M = 8	99.4787	380 *(10.145%)*	141 *(5.224%)*	381.018 *(0%)*	908.42 (0%)	298
M = 4	99.8527	403 (16.812%)	163 *(21.642%)*	381.018 (0%)	908.42 (0%)	312
[9]	-	204.651	179.675	-67.9	-67.9	-
[10]	99.293	21.75	1.5	-	-	-
[11]	92.82	-	-	-	-	-
[12]	-	102	-	612.208	1459.61	465.38
[13]	-	177	67	240	544.86	687

Compared to existing countermeasures, such as [9-11], the proposed scheme exhibits superior fault coverage of 99.8527%, exceeding the 99.293% and 92.82% achieved by [10] and [11], respectively. Although the area overhead of 16.812% and the increase in occupied slices of 21.642% are higher than those of some alternatives, the proposed scheme offers a competitive balance between robust fault detection and resource utilization, especially considering the maintained operating frequency. This makes it particularly suitable for resource-constrained IoT devices where a high level of security is crucial. The system in [9] shows promise in terms of FC efficiency, but further optimization may be needed to reduce the overhead of occupied slices without sacrificing performance. The design proposed in [12], although using the smallest area (102 slices LUTS), achieves the highest frequency (612.208 MHz) and throughput (1459.61 Mbps). This high performance comes at the cost of increased power consumption (465.38 mW). The implementation in [13] presents a more balanced profile, with an area between M= 8 and M= 4. Its frequency (240 MHz) and throughput (544.86 Mbps) are lower than [12] but still surpass M = 8 and M = 4. However, the design in [13] has the highest power consumption (687 mW) despite not having the highest performance. Table V also shows that M = 8 and M = 4 offer the lowest power consumption among all designs, but at the expense of lower frequency and throughput compared to [12]

and [13]. In general, the choice between the different implementations depends on the specific requirements of the application, balancing performance, area, frequency, throughput, and power consumption.

VI. CONCLUSION

This paper presented an enhanced fault detection scheme for the PRESENT lightweight block cipher, employing dynamic parity checking with varying multiplicity parameters ($M = 4$ and $M = 8$). The proposed scheme demonstrated a high level of effectiveness against both simple and multiple fault injection attacks, achieving a 99.8527% fault coverage rate. This enhanced security contributes to the development of more resilient infrastructure for critical systems, aligning with the goals of SDG 9 to build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation. The practicality of the proposed scheme based on parity makes it suitable for resource-constrained IoT devices deployed in such infrastructure.

The hardware implementation on a Xilinx Virtex5-XC5VFX70T FPGA platform demonstrated the practicality of the proposed scheme. While incurring a modest area overhead and an increase in occupied slices, the system maintained the original operating frequency. This represents a favorable trade-off between security and performance, making the scheme suitable for resource-constrained IoT devices. Compared to existing countermeasures, the scheme exhibits superior fault coverage while maintaining competitive resource utilization.

ACKNOWLEDGMENT

The authors extend their appreciation to Prince Sattam bin Abdulaziz University for funding this research work through the project number (PSAU/2024/01/31267).

REFERENCES

[1] A. Bogdanov *et al.*, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, 2007, pp. 450–466, https://doi.org/10.1007/978-3-540-74735-2_31.

[2] J. Jebrane and S. Lazaar, "A performance comparison of lightweight cryptographic algorithms suitable for IoT transmissions," *General Letters in Mathematics*, vol. 10, no. 2, pp. 46–53, Jun. 2021, https://doi.org/10.31559/glm2021.10.2.5.

[3] A. Kavitha *et al.*, "A Novel Algorithm to Secure Data in New Generation Health Care System from Cyber Attacks Using IoT," *International Journal of Electrical and Electronics Research*, vol. 10, no. 2, pp. 270–275, Jun. 2022, https://doi.org/10.37391/ijeer.100236.

[4] S. Sheikhpour, A. Mahani, and N. Bagheri, "Reliable advanced encryption standard hardware implementation: 32- bit and 64-bit data-paths," *Microprocessors and Microsystems*, vol. 81, Mar. 2021, Art. no. 103740, https://doi.org/10.1016/j.micpro.2020.103740.

[5] A. Jain and U. Guin, "A Novel Tampering Attack on AES Cores with Hardware Trojans," in *2020 IEEE International Test Conference in Asia (ITC-Asia)*, Taipei, Taiwan, Sep. 2020, pp. 77–82, https://doi.org/10.1109/ITC-Asia51099.2020.00025.

[6] H. Kwon, Y. B. Kim, S. C. Seo, and H. Seo, "High-Speed Implementation of PRESENT on AVR Microcontroller," *Mathematics*, vol. 9, no. 4, Jan. 2021, Art. no. 374, https://doi.org/10.3390/math9040374.

[7] K. Keerthi and C. Rebeiro, "FaultMeter: Quantitative Fault Attack Assessment of Block Cipher Software," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 212–240, Mar. 2023, https://doi.org/10.46586/tches.v2023.i2.212-240.

[8] H. Mestiri, I. Barraj, T. Saidani, and M. Machhout, "A PRESENT Lightweight Algorithm High-Level SystemC Modeling using AOP Approach," *Engineering, Technology & Applied Science Research*, vol. 14, no. 5, pp. 16772–16777, Oct. 2024, https://doi.org/10.48084/etasr.8417.

[9] S. Patranabis *et al.*, "Lightweight Design-for-Security Strategies for Combined Countermeasures Against Side Channel and Fault Analysis in IoT Applications," *Journal of Hardware and Systems Security*, vol. 3, no. 2, pp. 103–131, Jun. 2019, https://doi.org/10.1007/s41635-018-0049-y.

[10] J. Breier, W. He, D. Jap, S. Bhasin, and A. Chattopadhyay, "Attacks in Reality: the Limits of Concurrent Error Detection Codes Against Laser Fault Injection," *Journal of Hardware and Systems Security*, vol. 1, no. 4, pp. 298–310, Dec. 2017, https://doi.org/10.1007/s41635-017-0020-3.

[11] W. He, J. Breier, S. Bhasin, N. Miura, and M. Nagata, "Ring Oscillator under Laser: Potential of PLL-based Countermeasure against Laser Fault Injection," in *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Santa Barbara, CA, USA, Aug. 2016, pp. 102–113, https://doi.org/10.1109/FDTC.2016.13.

[12] T. Kowsalya, R. Ganesh Babu, B. D. Parameshachari, A. Nayyar, and R. Majid Mehmood, "Low Area PRESENT Cryptography in FPGA Using TRNG-PRNG Key Generation," *Computers, Materials & Continua*, vol. 68, no. 2, pp. 1447–1465, 2021, https://doi.org/10.32604/cmc.2021.014606.

[13] J. G. Pandey, T. Goel, and A. Karmakar, "Hardware architectures for PRESENT block cipher and their FPGA implementations," *IET Circuits, Devices & Systems*, vol. 13, no. 7, pp. 958–969, 2019, https://doi.org/10.1049/iet-cds.2018.5273.

[14] M. Zhang, H. Li, P. Wang, and Q. Liu, "Parity Check Based Fault Detection against Timing Fault Injection Attacks," *Electronics*, vol. 11, no. 24, Jan. 2022, Art. no. 4082, https://doi.org/10.3390/electronics11244082.

[15] H. Mestiri and I. Barraj, "High-Speed Hardware Architecture Based on Error Detection for KECCAK," *Micromachines*, vol. 14, no. 6, Jun. 2023, Art. no. 1129, https://doi.org/10.3390/mi14061129.

[16] H. Mestiri, N. Benhadjyoussef, and M. Machhout, "Fault Attacks Resistant AES Hardware Implementation," in *2019 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS)*, Gammarth-Tunis, Tunisia, Apr. 2019, pp. 1–6, https://doi.org/10.1109/DTSS.2019.8914979.