# Classic McEliece Hardware Implementation with Enhanced Side-Channel and Fault Resistance

Peizhou Gan, Prasanna Ravi, Kamal Raj, Anubhab Baksi* and Anupam Chattopadhyay
Nanyang Technological University, Singapore
{peizhou.gan, prasanna.ravi, kamal.raj, anubhab.baksi, anupam}@ntu.edu.sg

***Abstract*—In this work, we propose the first hardware implementation of Classic McEliece protected with countermeasures against Side-Channel Attacks (SCA) and Fault Injection Attacks (FIA). Classic Mceliece is one of the leading candidates for Key Encapsulation Mechanisms (KEMs) in the ongoing round 4 of the NIST standardization process for post-quantum cryptography. In particular, we implement a range of generic countermeasures against SCA and FIA, particularly protected the vulnerable operations such as additive Fast Fourier Transform (FFT) and Gaussian elimination, that have been targeted by prior SCA and FIA attacks. We also perform a detailed SCA evaluation demonstrating no leakage even with 100000 traces (improvement of more than $100\times$ the number of traces compared to unprotected implementation). This comes at a modest total area overhead of between $4\times$ to $7\times$, depending on the type of implemented SCA countermeasure. Furthermore, we present a thorough ASIC benchmark for SCA and FIA protected Classic McEliece design.**

***Index Terms*—Post-quantum Cryptography, Classic McEliece, Side-Channel Attack, Fault Injection Attack, Countermeasures, ASIC**

## I. Introduction

PUBLIC key cryptosystems are an integral part in our current society as it impacts virtually every possible digital communication where there is a need to establish secure communication channels. Widely used key exchange algorithms, such as RSA or ECC, rely on some hardness assumption (like integer factorization). However, those are threatened by the upcoming quantum computers. To overcome this issue, a relatively new class of ciphers (called, post-quantum cryptography) have been proposed which can withstand the quantum threat.

Classic McEliece is one prominent post-quantum cipher. It is one of the candidates for KEMs from code-based cryptography in the on-going fourth round of the NIST PQC standardization process. The security of Classic McEliece is derived from the hardness assumption of decoding a linear code [1]. Over the years, there have been many Side-Channel (SCA) and Fault Injection (FIA) attacks on Classic Mceliece Key Encapsulation Mechanism (KEM) [2]–[8]. Particularly, the attackers have targeted the additive Fast Fourier Transform (FFT) and Gaussian elimination operations, which are also more susceptible to SCA and FIA. SCA countermeasures are necessary in Classic McEliece, which already utilizes large area in hardware. Thus, it is more attractive for a designer to consider low cost generic countermeasures to protect Classic McEliece against SCA, instead of masking, which will incorporate plenty of area and performance overhead. Traditionally, SCA and FIA were considered as separated attack methodologies. However, combined attack of SCA and FIA on RSA exponentiation is shown in [9], while a combined protection proposed in [10] to resist combined attack.

The contributions in this paper can be stated as follows:

- In this work, the first hardware implementation of the Classic McEliece KEM protected against both SCA and FIA is proposed, which particularly emphasize on protecting the vulnerable operations, namely, the additive FFT and the Gaussian elimination.
- A Gaussian Noise Generator and a Randomized Clock against SCA are implemented, while Random Delay Insertion and Redundancy are used to protect against FIA.
- In particular, we perform SCA evaluation using standard Test Vector Leakage Analysis (TVLA) [11], and were able to validate the absence of side-channel leakage for up to 100000 traces, with an increase in execution time (avg. clock cycles) between $20\%$ and $80\%$.
- To the best of our knowledge, we perform the first ASIC benchmark for the SCA and FIA protected Classic McEliece hardware design.

The rest of this paper is organized as follows. First, a briefly description of Classic McEliece (Section II) is provided. Then, existing FIA and SCA designs are described. In Section III and Section IV, the countermeasures and ASIC benchmarks are introduced respectively. The experimental results are summarized in Section V. Finally, the conclusion is presented in Section VI. Further, other information (such as, SCA traces) are shared online[1].

## II. Preliminaries on Classic McEliece

### A. Concise Algorithmic Description

The code-based McEliece cryptosystem was first proposed in 1978 based on randomly chosen irreducible Goppa codes, and in 1986, a dual variant of McEliece cryptosystem was introduced by Niederreiter [12], that uses a parity check matrix $H$ (instead of a generator matrix $G$) to encrypt a message $m$ into an error vector $e$. The Classic McEliece algorithm, submitted as part of the NIST PQC standardization process is

*: This project is partially supported by the Wallenberg-NTU Presidential Post-Doctoral Fellowship.

[1] https://github.com/gan-pz/mceliece-figures/

based on the Niederreiter framework. It is essentially an IND-CCA2 secure KEM, obtained from transforming an IND-CPA secure Niederreiter encryption scheme.

The following provides a brief introduction to the three main procedures of the Classic McEliece KEM: key generation, encapsulation, and decapsulation. The adjustable input parameters for a Classic McEliece KEM are:

- $n$: Code length of the Goppa code used.
- $m$: Size of the finite field used $q$ is defined by $q = 2^m$.
- $t$: Number of correctable errors for the Goppa Code.
- Therefore, the code dimension, $k = n - mt$.

*1) Key Generation:* The key generation procedure generates a public and private key pair based on the input parameters in the manner shown in Algorithm 1.

**Algorithm 1** Classic McEliece Key-pair Generation

**Input:** The Classic McEliece parameter set: $m, t, n$
**Output:** Secret Key $(g(x), L)$, Public Key (T)

1: Generate a uniform random $n$-bit string $s$
2: Generate a uniform random monic irreducible polynomial $g(x) \in \mathbb{F}_{\shortparallel}[x]$ of degree $t$
3: Select a uniform random sequence $L = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ of $n$ distinct elements in $\mathbb{F}_{\shortparallel}$, as the support for Goppa code
4: Compute the parity check matrix $H$ of size $(t \times n)$ for the Goppa Code $\Gamma$ defined by $(g(x), L)$
5: Form the $(mt \times n)$ binary matrix $H'$ by replacing each entry in $H$ with a column of $m$ bits, where the bits correspond to the coefficient of polynomial representation of the entry
6: Reduce $H'$ to systematic form $H'' = [I_{n-k}|T]$, where $I_{n-k}$ is an $(n-k) \times (n-k)$ identity matrix

*2) Encapsulation:* The Encapsulation algorithm is used to generate a ciphertext and a session key $K$, where K is generated using the encoding algorithm and the cryptographic hash function $h$, as shown in Algorithm 2.

**Algorithm 2** Classic McEliece Encapsulation

**Input:** Public Key $(T)$
**Output:** Session Key $(K)$, Ciphertext $(C)$

1: Generate a uniform error vector $e \in \mathbb{F}_2^n$ of weight $t$
2: $C_0 \leftarrow He \in \mathbb{F}_2^{n-k}$, where $H = [I_{n-k}|T]$
3: $C_1 \leftarrow h(2, e)$ and put $C \leftarrow (C_0, C_1)$
4: $K \leftarrow h(1, e, C)$

*3) Decapsulation:* With the input ciphertext $C$, the session key $(K)$ is computed in decapsulation, as described in Algorithm 3.

**Algorithm 3** Classic McEliece Decapsulation

**Input:** Ciphertext $(C)$, Secret Key $(g(x), L)$
**Output:** Session Key $(K)$

1: Split ciphertext $C$ as $(C_0, C_1)$ and set $b = 1$
2: Extract $s \in \mathbb{F}_2^n$ and $\Gamma'(g(x), L')$ from secret key
3: Extend $C_0$ to $v = (C_0, 0, \ldots, 0) \in \mathbb{F}_2^n$ by inserting $k$ 0's
4: Find the unique codeword $c$ in the Goppa code defined by $\Gamma'$ that is at distance $\leq t$ from $v$
5: Set $e = v + c$
6: If $wt(e) = t$, $C_0 = He$, return $e$; otherwise return $\perp$, where $H = [I_{n-k}|T]$
7: If $e \leftarrow \perp$, set $e \leftarrow s$ and $b \leftarrow 0$
8: $C_1' \leftarrow h(2, e)$
9: If $C_1' \neq C_1$ set $e \leftarrow s$ and $b \leftarrow 0$
10: $K \leftarrow h(b, e, C)$

## B. Prior Side-Channel and Fault Injection Attacks

*1) SCA on Key generation:* [8] proposed a key recovery attacks by utilizing the power leakage during Gaussian elimination (Algorithm 1, line 6) in public key generation procession.

*2) SCA and FIA on Encapsulation:* Aiming at matrix vector multiplication during encapsulation (Algorithm 2, line 2), message recovery attacks were implemented in [3] using laser FIA, and in [5] using power attacks respectively.

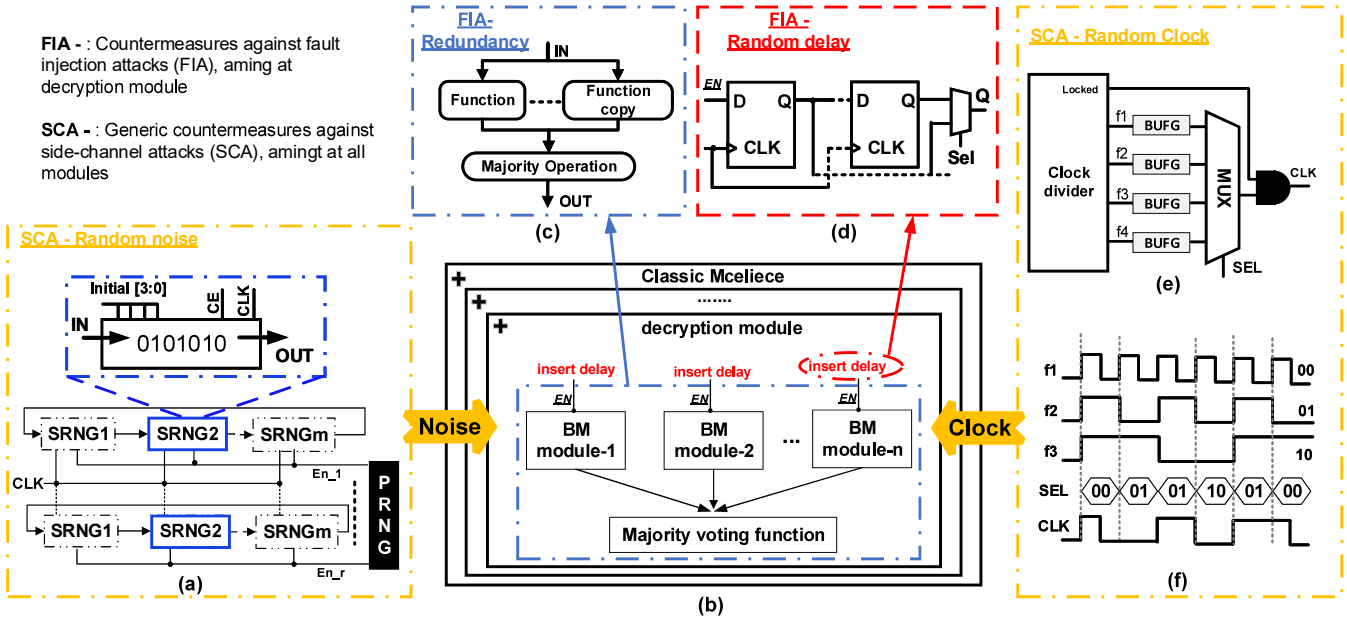
*3) SCA and FIA on Decapsulation:* Based on the information that the commonly used additive FFT in the decryption module (Algorithm 3, line 4) will cause potential leakage, [2], [6] proposed the plaintext recovery and key recovery side-channel attack, respectively, together with power attacks on FPGA. Targeting the decryption in software design, [7] proposed the template syndrome decoding attack using the leakage from a Hamming-weight (HW) computation in post processing state of decryption. To be noticed, the protection for this kind of attack is not discussed in our paper because the HW computation is not included in the proposed hardware implementation. As for the FIA, [4] proposed an attack which targets the error-locator polynomial module (Berlekamp-Massey module) (Algorithm 3, line 4) and bypasses the validity checks operation (line 9 in Algorithm 3).

Differ from the deliberate fault detection designs [13]–[15], the proposed countermeasures is designed for general usage. To the best of our knowledge, the proposed hardware implementation is the first generic countermeasures protecting Classic McEliece hardware against both SCA and FIA. To be more precise, we protect the sensitive operations such as Gaussian elimination (key generation) and additive FFT (decapsulation), which leak the long-term secret key. Since the long-term secret key (key generation, decapsulation) is more sensitive compared to the session key (encapsulation), we focus on the implementing countermeasures only for the key-generation and decapsulation procedures, while our countermeasures can also be extended to the encapsulation procedure as well.

# III. Proposed FIA and SCA Countermeasures

## A. Proposed Generic SCA Countermeasures

*1) Proposed Gaussian Noise Generator:* It is well known that introducing random noise from additional circuits running in parallel with the cryptographic module reduces the signal-to-noise ratio (SNR) in power traces, thereby increasing the difficulty of being attakced. A shift register based noise generator (SRNG) (Figure 1a) rather than the shift register LUTs on [16] is proposed for generic usage but not only for FPGA implementation. In the implementation, the basic




(a) SCA-random noise; (b) Overview of proposed countermeasures; (c) FIA-Redundancy; (d) FIA-Random delay; (e) SCA-Random clock; (f) SCA-Random clock timing diagram.

**Fig. 1:** Overview of proposed countermeasures, including FIA and SCA countermeasures.

element of SRNG is a chain of 64 16-bits shift registers connected end to end. When the enable signal CE is set to high, the initial value in shift register keeps shifting from LSB to MSB. The SRNG consists of $r \times m$ shift registers ($r = 64, m = 16$ in this design), with the enable signal CE controlled by a linear feedback shift register (LSFR) based PRNG. The parameters $r$ and $m$ are adjustable to control the amount of noise introduced to the side-channel traces.

*2) Proposed Randomized Clock:* To achieve the generic usage for both FPGA and ASIC, a dedicated clock divider circuit rather than the Mixed-Mode Clock Management (MMCM) block [17] is introduced as a randomized clock generator in the proposed countermeasure , as show in Figure 1e. A tree of multiplexers is then designed to select one of the frequencies randomly. In the TVLA test, four different clocks (40MHz, 20MHz, 10MHz, 5MHz) are generated.

The generic nature of these countermeasures enable them to be applied for the entire Classic McEliece design. In the proposed implementation, the security of the two generic countermeasures are evaluated on the most vulnerable operations additive FFT and Gaussian elimination module as show in Section V.

### B. Proposed Generic FIA Countermeasures

*1) Redundancy:* The hardware redundancy countermeasure works by duplicating the key circuit and comparing the results to check for discrepancies. Figure 1c illustrates the hardware redundancy approach [18]. While redundancy is straightforward and effective, it comes with a high resource cost.

*2) Random Delay Insertion:* The main idea of this countermeasure is to induce random delays on the input signals of ctitial pipeline stages of the processor data path, resulting in a randomization of the charge quantity from the power supply. A D flip-flop with a random wait state (RWDFF) [19] is implemented, which can insert multiple random clock period delays into the data path, shown in Figure 1d.

In this work, FIA countermeasures consist of redundancy and random delay techniques, as shown in Figure 1b. The redundancy is implemented by creating three copies of the Berlekamp-Massey (BM) decoder module from the decapsulation process. A majority voting mechanism is then applied to the outputs of the BM modules, protecting the validity check (line 9 in Algorithm 3) against the fault injection attack described in [4]. Additionally, a random delay, is introduced to the enable input signal EN of BM modules.

## IV. Classic McEliece ASIC Benchmarks

An ASIC implementation is distinguished from FPGA-based hardware designs, providing significantly better runtime, energy and area performances. The RTL source code for Classic McEliece is revised from the FPGA implementation in [20] to ASIC by resolving some synthesizing errors that can be ignored by the FPGA but not the ASIC design flow. Apart from that, the proposed SCA and FIA countermeasures are implemented. We utilize the *Synopsys Design Compiler* (S-2021.06-SP1) for synthesis and *Cadence Innovous* (version v21.17-s075_1) for place and route (PNR), and use the well-established TSMC 65nm CMOS (TCBN65LP) technology to perform ASIC benchmarking of our Classic McEliece design.

The ASIC results are based on the Single-Pass Early-Abort (SPEA) systemizer from [19]. External memory is used for the testbench due to the large public key size in Classic McEliece and the lack of an embedded commercial SRAM Compiler IP. Table I compares the operating frequency of our Kintex-7 XC7K325T-2FFG900C FPGA designs with the ASIC designs for all Classic McEliece parameter sets. Overall, the ASIC

**TABLE I:** Classic McEliece implementation comparison between ASIC and FPGA

| Parameter Set | Freq. (MHz) @ASIC | | | Freq. (MHz) @FPGA | | |
|---|---|---|---|---|---|---|
| | Key gen.[1] | Encap. | Decap. | Key gen.[1] | Encap. | Decap. |
| CM348864 | 387 | 416 | 357 | 158 | 172 | 169 |
| CM460896 | 387 | 416 | 357 | 158 | 172 | 169 |
| CM6688128 | 387 | 416 | 357 | 150 | 172 | 169 |
| CM6960119 | 387 | 416 | 357 | 150 | 172 | 169 |
| CM8192128 | 387 | 416 | 357 | 150 | 172 | 169 |

1: Using peripheral/external memory block

shows improved performance, with key generation being about 2.2× faster, encapsulation 2.4× faster, and decapsulation 2.1× faster than the FPGA implementations.

## V. Experimental Evaluation

In this part, we report the experimental evaluation of our SCA countermeasures for Classic McEliece. Our designs are tested on the SAKURA-X board with a Xilinx Kintex-7 XC7K160T-1FBGC FPGA[2].

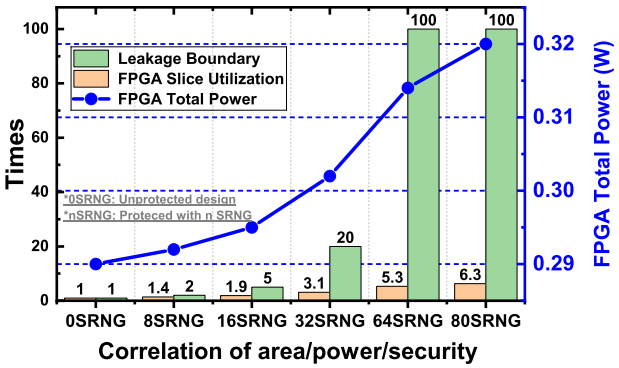



**Fig. 2:** Correlation at additive FFT module

### A. SCA using TVLA with/without Countermeasure

To evaluate the security of our generic SCA countermeasures on the Classic McEliece design, the well-known Test Vector Leakage Analysis (TVLA) [11] are adopted. In particular, we utilize the non-specific fixed versus random t-test, with the TVLA threshold value of ±4.5 indicating presence of leakage.

In practice, the evaluation of the countermeasures focuses on the additive FFT and Gaussian elimination modules, which are attacked in [6], [7]. As illustrated in Figure 2, prior to conducting the TVLA test, we first examine the relationship between resources (FPGA Slice Utilization), power consumption (FPGA Total Power), and security (Leakage Boundary) across various security levels. This is achieved by modifying the amount of SRNG used in the countermeasures. Taking the additive FFT module as a reference for analysis, we consider the unprotected design, labeled as "0SRNG" in Figure 2, as the baseline for both resource and security evaluations. FPGA total power is estimated using Vivado Simulator SAIF data following post-implementation timing simulations. Moreover, the unprotected design shows leakage points under TVLA tests within only 1000 power traces (Figure 3). From the observed correlation, the countermeasures enhance security by at least 100×, albeit with a 5× overhead in resource consumption.

[2]http://www.meytang.com/h-pd-20.html

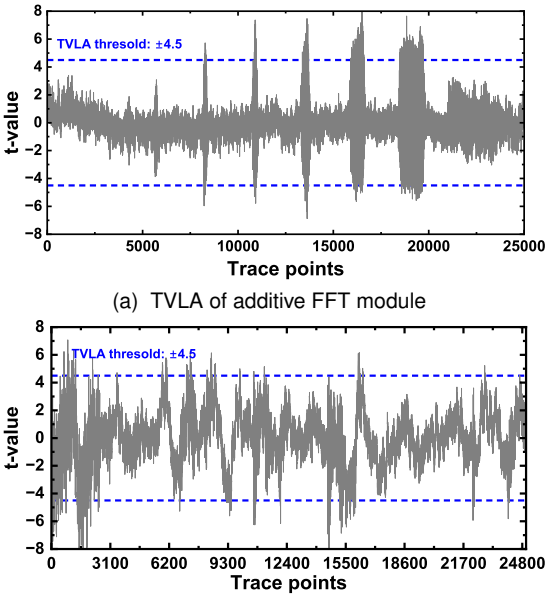



**Fig. 3:** TVLA results on unprotected design within 1000 power traces

Figure 4a and Figure 4b display the TVLA plots for the protected designs, which evaluate the protected implementation with a combination of random clocking and noise (64 SRNGs) countermeasures. These plots, generated from 100,000 power traces, showing no observable leakage, demonstrating a significant improvement in SCA resilience. This confirms the effectiveness of our generic SCA countermeasures in protecting the Classic McEliece hardware.

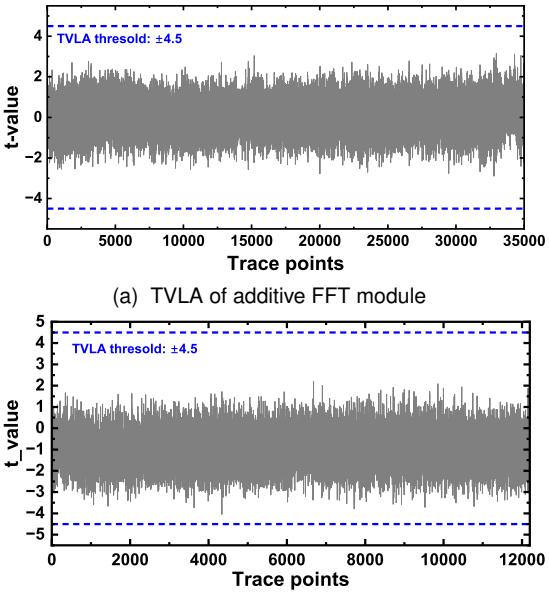



**Fig. 4:** TVLA results on protected design within 100000 power traces

### B. Performance Overhead from SCA and FIA Countermeasures

Refer to Table II for the performance overhead (area and clock cycle count) due to our generic SCA and FIA countermeasures for the Classic McEliece design on FPGA and ASIC respec-

tively. Note that, we utilize the CM348864 as a representative parameter set set of Classic McEliece for illustration.

Regarding the area overhead, the slice utilization of the additive FFT and Gaussian elimination module rise between $5\times$ and $10\times$ in the FPGA benchmarks, while the ASIC indicate a $4\times$ to $7\times$ increase for the additive FFT and Gaussian elimination, respectively. The significant area overhead results from two main reasons: 1) In the FPGA benchmarks, the countermeasures require more slice resources and less BRAM due to its structural characteristics. In addition, a unified measurement standard between slice and BRAM resources is absent on FPGA platforms. Thus, compared to the overhead on FPGA benchmark, the ASIC benchmark is more accurate. 2) Compared to the substantial area requirements of the entire Classic McEliece design, the resources required for the countermeasures are negligible. Specifically, the area of the countermeasures accounts less than 24% of the entire decryption module, let alone the whole Classic McEliece design.

As for the average clock cycle count, we observe an increase between $20\%$ to $80\%$ due to the SCA countermeasures in both FPGA and ASIC benchmarks. Thus, we observe that our generic shared SCA countermeasures can be implemented with a reasonable overhead in performance and runtime for Classic Mceliece, given that Classic Mceliece is mainly intended for high-security environments where resource consumption is typically not a hard constraint.

For the FIA countermeasures, compared with the unprotected decryption module, we observe an increase in area consumption of roughly $1.6\times$ and $1.2\times$ in FPGA and ASIC benchmarks respectively, which therefore also clearly demonstrates a reasonable overhead for our FIA protected design.

**TABLE II:** Area and Performance Comparison of the unprotected and protected Classic McEliece hardware design

| Tested Modules | Category | FPGA benchmarks | | ASIC benchmarks | | Avg. clock cycles |
|---|---|---|---|---|---|---|
| | | SLICE (%) | BRAM (%) | Cell Count | Area ($\mu$m) | |
| Decryption | w/o FIA- [1] | 14.2 | 4.8 | 330063 | 1281349 | 1806 |
| | w/ FIA- [2] | 23.1 | 4.9 | 407164 | 1547007 | 1806 |
| additive FFT | w/o SCA- [3] | 2.9 | 0 | 19906 | 83190 | 1095 |
| | w/ SCA- [4] | 15.3 | 0 | 57052 | 368399 | 1996 |
| Gaussian elimination | w/o SCA- | 1.0 | 0 | 8294 | 45053 | 160 |
| | w/ SCA- | 10.1 | 0 | 45479 | 330275 | 193 |

1: Without FIA countermeasures; 2: With FIA countermeasures; 3: Without SCA countermeasures; 4: With SCA countermeasures.

## VI. Conclusion

In this paper, we present the first hardware implementation of Classic McEliece protected with generic side-channel analysis (SCA) and fault injection attack (FIA) countermeasures. Specifically, we secure vulnerable operations such as the additive FFT and Gaussian elimination modules, which have been the primary targets of previous attacks on Classic McEliece. The propsoed practical SCA evaluation shows no leakage even after 100,000 power traces—a significant improvement compared to the leakage observed at just 1,000 traces in the unprotected design. This is achieved with reasonable area and performance overheads, demonstrating the effectiveness of our protected design.

## References


[1] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[2] N. Lahr, R. Niederhagen, R. Petri, and S. Samardjiska, "Side channel information set decoding using iterative chunking," Cryptology ePrint Archive, Paper 2019/1459, 2019, https://eprint.iacr.org/2019/1459.

[3] P.-L. Cayrel, B. Colombier, V.-F. Drăgoi, A. Menu, and L. Bossuet, "Message-recovery laser fault injection attack on the classic mceliece cryptosystem," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 438–467.

[4] S. Pircher, J. Geier, J. Danner, D. Mueller-Gritschneder, and A. Wachter-Zeh, "Key-recovery fault injection attack on the classic mceliece kem," Cryptology ePrint Archive, Paper 2022/1529, 2022, https://eprint.iacr.org/2022/1529.

[5] B. Colombier, V.-F. Drăgoi, P.-L. Cayrel, and V. Grosso, "Profiled side-channel attack on cryptosystems based on the binary syndrome decoding problem," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3407–3420, 2022.

[6] Q. Guo, A. Johansson, and T. Johansson, "A key-recovery side-channel attack on classic mceliece implementations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 4, p. 800–827, Aug. 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9841

[7] S. Bitzer, J. Delvaux, E. Kirshanova, S. Maaßen, A. May, and A. Wachter-Zeh, "How to lose some weight-a practical template syndrome decoding attack," *Cryptology ePrint Archive*, 2024.

[8] M. Brinkmann, C. Chuengsatiansup, A. May, J. Nowakowski, and Y. Yarom, "Leaky mceliece: Secret key recovery from highly erroneous side-channel information," *Cryptology ePrint Archive*, 2023.

[9] F. Amiel, K. Villegas, B. Feix, and L. Marcel, "Passive and active combined attacks: Combining fault attacks and side channel analysis," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2007)*. IEEE, 2007, pp. 92–102.

[10] T. Schneider, A. Moradi, and T. Güneysu, "Parti–towards combined hardware countermeasures against side-channel and fault-injection attacks," in *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36*. Springer, 2016, pp. 302–332.

[11] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi *et al.*, "Test vector leakage assessment (tvla) methodology in practice," in *International Cryptographic Module Conference*, vol. 1001. sn, 2013, p. 13.

[12] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Prob. Contr. Inform. Theory*, vol. 15, no. 2, pp. 157–166, 1986.

[13] A. Cintas-Canto, M. M. Kermani, and R. Azarderakhsh, "Reliable architectures for finite field multipliers using cyclic codes on fpga utilized in classic and post-quantum cryptography," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 1, pp. 157–161, 2022.

[14] A. C. Canto, M. M. Kermani, and R. Azarderakhsh, "Reliable constructions for the key generator of code-based post-quantum cryptosystems on fpga," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 19, no. 1, pp. 1–20, 2022.

[15] A. Cintas-Canto, M. Mozaffari-Kermani, R. Azarderakhsh, and K. Gaj, "Crc-oriented error detection architectures of post-quantum cryptography niederreiter key generator on fpga," in *2022 IEEE Nordic Circuits and Systems Conference (NorCAS)*. IEEE, 2022, pp. 1–7.

[16] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2011, pp. 33–48.

[17] M. Brisfors, M. Moraitis, and E. Dubrova, "Side-channel attack countermeasures based on clock randomization have a fundamental flaw," *Cryptology ePrint Archive*, 2022.

[18] F. E. Potestad-Ordóñez, E. Tena-Sánchez, A. J. Acosta-Jiménez, C. J. Jiménez-Fernández, and R. Chaves, "Hardware countermeasures benchmarking against fault attacks," *Applied Sciences*, vol. 12, no. 5, p. 2443, 2022.

[19] M. Bucci, R. Luzzi, M. Guglielmo, and A. Trifiletti, "A countermeasure against differential power analysis based on random delay insertion," in *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2005, pp. 3547–3550.

[20] P.-J. Chen, T. Chou, S. Deshpande, N. Lahr, R. Niederhagen, J. Szefer, and W. Wang, "Complete and improved fpga implementation of classic mceliece," Cryptology ePrint Archive, Paper 2022/412, 2022, https://eprint.iacr.org/2022/412.