A Side-Channel Attack against Classic McEliece when loading the Goppa Polynomial.

Boly Seck, Pierre-Louis Cayrel, Vlad-Florin Dragoi, Idy Diop, Morgan Barbier, Jean Belo Klamti, Vincent Grosso, Brice Colombier

A Side-Channel Attack against *Classic McEliece* when loading the Goppa Polynomial

Boly Seck[1,2][0000−0002−2362−601X], Pierre-Louis Cayrel[2][0000−0002−6708−868X], Vlad-Florin Dragoi[3][0000−0002−8673−9097], Idy Diop[1][0000−0002−9143−196X], Morgan Barbier[4], Jean Belo Klamti[5][0000−0001−9231−1129], Vincent Grosso[2][0000−0002−3874−7527], and Brice Colombier[2][0000−0002−6028−3028]

[1] ESP, Laboratoire d'imagerie médicale et de Bio-informatique, Dakar, Sénégal
seck.boly@ugb.edu.sn
idy.diop@esp.sn

[2] Univ Lyon, UJM-Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France
{pierre.louis.cayrel,vincent.grosso,b.colombier}@univ-st-etienne.fr

[3] Faculty of Exact Sciences, Aurel Vlaicu University, Arad, Romania
vlad.dragoi@uav.ro

[4] ENSICAEN, Groupe de recherche en informatique et instrumentation de Caen, CNRS, Boulevard Maréchal Juin 14 000, Caen, France
morgan.barbier@ensicaen.fr

[5] Department of Electrical & Computer Engineering, University of Waterloo
jbklamti@uwaterloo.ca

Abstract. The NIST Post-Quantum Cryptography (PQC) standardization challenge was launched in December 2016 and recently, has released its first results. The whole process has given a considerable dynamic to the research in post-quantum cryptography, in particular to practical aspects, such as the study of the vulnerabilities of post-quantum algorithms to side-channel attacks. In this paper, we present a realistic template attack against the reference implementation of *Classic McEliece* which is a finalist of the 4th round of NIST PQC standardization. This profiled attack allowed us to accurately find the Hamming weight of each coefficient of the Goppa polynomial. With only one decryption, this result enables us first, to find directly the Goppa polynomial in the case of weak keys with the method of Loidreau and Sendrier (P. Loidreau and N. Sendrier, "Weak keys in the McEliece public-key cryptosystem", *IEEE Trans. Inf. Theory*, 2001). Then, in the case of "slightly less weak keys", we also find this polynomial with an exhaustive search with low complexity. Finally, we propose the best complexity reduction for exhaustive Goppa polynomial search on $\mathbb{F}_{2^m}$. We attack the constant-time implementation of *Classic McEliece* proposed by Chen *et al.*. This implementation, which follows the NIST specification, is realized on a `stm32f4-Discovery` microcontroller with a 32-bit ARM Cortex-M4.

Keywords: NIST PQC standardization· Classic McEliece· Side-Channel Attack· Template Attack· Goppa Polynomial

1 Introduction

In recent years, research on quantum computers has accelerated considerably [TF19; GI19; Lar+21]. These computers can theoretically solve difficult number theory problems (the integer factorization problem and the discrete logarithm problem) in polynomial time [Fey18; DJ92; Gro96; Sho94]. Thus, if large-scale quantum computers are built, they will be able to break most current asymmetric systems such as RSA, ECDSA and ECDH. This would severely compromise the confidentiality and integrity of all digital communications. As a result, the cryptographic community has turned its attention to credible alternatives for dealing with quantum computing. Thus, in 2016, the National Institute of Standards and Technology (NIST) announced a call for proposals to standardize post-quantum cryptography (PQC) primitives [CML17]. This standardization process consists of several rounds, and only those applicants that best meet NIST's requirements in each round are selected to proceed to the next round.

On the fifth of July, 2022, NIST released the first four winning algorithms (a key establishment algorithm named CRYSTALS-Kyber, and three digital signature algorithms named CRYSTALS-Dilithium, FALCON, and SPHINCS+). The first three of these algorithms are based on structured lattices and the last one, SPHINCS+ is a hash-based signature scheme. These future standards are expected to be used by default for selecting post-quantum algorithms in many security products. Provided that these post-quantum algorithms are also combined with proven classical algorithms through hybrid mechanisms. The main goal of the process started by NIST is to replace three standards that are considered the most vulnerable to quantum attacks, *i.e.*, FIPS 186-4[6] (for digital signatures), NIST SP 800-56A[7], and NIST SP 800-56B[8](both for keys establishment in public-key cryptography).

Beside the four winners, an extension of the NIST PQC standardization campaign (4th round) is planned for key establishment algorithms: BIKE, HQC, *Classic McEliece* [Cho+20] (all three based on error-correcting codes). *Classic McEliece* was the first selected finalist for code-based cryptography as a Key Encapsulation Mechanism (KEM), while BIKE and HQC were two alternatives.

In addition to defining secure schemes and choosing secure parameters, an important issue, in the standardization process, is the impact of a scheme's implementation on its security. A general requirement on the implementation of a scheme is that the execution time of operations does not vary with the secret information (e.g., the secret key or plaintext). This is called a constant-time implementation. However, there are other side-channel attacks beside timing attacks that can allow an attacker to access secret information. Other side-channels include power consumption and electromagnetic, photonic, and acoustic emissions.

[6] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.PDF
[7] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.PDF
[8] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br1.PDF

For many PQC systems, it is still unclear which side-channel attacks are feasible in practice and how to be protected against them. That is why the side-channel topic has become recurent in the NIST PQC seminar[9] [Saa22; Rav+22]. In the past, code-based cryptosystems were subject to side-channel attacks even before NIST started the standardisation process [CD10; HMP10; Mol+11; Ava+11; Che+16]. However, the end of the first round of this challenging process defined the beggining of a side-channel race for physical security assesment. Indeed, *Classic McEliece* KEM oriented attacks appeared in a series of articles, where either the security of session key or of the private key was investigated [Lah+20; Cay+21; GJJ22; Col+22a; Sec+22; Gro+23; Col+23]. Here we make another step in this direction.

Contribution: In this work, we focus on the constant-time implementation of *Classic McEliece* which is one of the finalists of the NIST PQC extended campaign. This is a KEM, which is conservatively built from a Public Key Encryption (PKE) designed for One-way under Chosen-Plaintext Attack (OW-CPA) security, namely Niederreiter's dual version of McEliece's PKE using Goppa binary codes, as described in Section 2.2. First, we perform a template attack during decryption in *Classic McEliece* to find the Hamming weights of the Goppa polynomial coefficients. Since the Goppa polynomial is loaded from memory during this step, we were able to track the execution step of the algorithm and measure the corresponding power consumption. With this information at hand, we have built a profile for each possible weight and deployed, using a single trace, our attack. At the end of this step, each weight was detected with an almost perfect accuracy. Finally, we have used this information to find the Goppa polynomial. In the case of weak keys with binary coefficients, the polynomial was retrieved directly from the Hamming weight. In the case of "slightly less weak keys", we show that in polynomial time one can compute the Goppa polynomial. Moreover, we have significantly improved the complexity of the best attack to find the Goppa polynomial on $\mathbb{F}_{2^m}$ from 2^{1615} to 2^{1174} for $m = 13$, $n = 8192$ and $t = 128$. Finally, we show that our attack is realistic compared to other side-channel attacks on *Classic McEliece*.

Organization: The paper is organized as follows: Section 2 provides some background information on code-based cryptography and briefly describes *Classic McEliece* scheme. Section 3 presents a detailed description of our template attack on the constant-time implementation of *Classic McEliece* on ARM Cortex-M4 [CC21] to recover the Hamming weights of the Goppa polynomial coefficients. In Section 4, we use this result to find the Goppa polynomial directly in the case of weak keys and reduce the complexity of the exhaustive search for the Goppa polynomial on $\mathbb{F}_{2^m}$. Section 5 compares our attack with a recent side-channel attack of Guo *et al.* [GJJ22] on *Classic McEliece* and finally, we conclude this paper in Section 6.

[9] https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline/pqc-seminars

2 Theoretical background

2.1 Preliminaries

Let us first recall basic definitions and notations used in coding theory. A linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a vector subspace of $\mathbb{F}_q^n$ of dimension k and the elements of $\mathcal{C}$ are called codewords. A generator matrix of $\mathcal{C}$ is a matrix $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$ such that its lines form a basis of the vector space $\mathcal{C}$ such that $\mathcal{C} = \{\boldsymbol{xG} | \boldsymbol{x} \in \mathbb{F}_q^k\}$. A parity-Check matrix is a matrix $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$ such that $\mathcal{C} = \{\boldsymbol{y} \in \mathbb{F}_q^n | \boldsymbol{Hy}^T = 0\}$.

To perform error detection and correction, a code $\mathcal{C}$ uses a norm, the most common one being the Hamming weight $wt(\boldsymbol{y}) = \#\{i, \boldsymbol{y}_i \neq 0\}$. Any linear code possesses a minimum distance, i.e.,

$$d(\mathcal{C}) = \min\{wt(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathcal{C}, \boldsymbol{c} \neq 0\}. \tag{1}$$

The majority of structured codes with Hamming distance d can correct any error $\boldsymbol{e}$ of Hamming weight $wt(\boldsymbol{e}) \leq \lfloor (d-1)/2 \rfloor$. This quantity, which we will refer to as t in the following, refers to the correction capacity of $\mathcal{C}$.

Several families of codes were proposed in the literature as possible solutions in a cryptographic context such as the McEliece cryptosystem [Sid94; Nie86; SK14; Mis+13]. However, many of these were completely broken [CB14; Cou+14; Bar+16; OK15]. The original proposal by McEliece that uses binary Goppa codes remains unbroken. Thus, we will remind here some properties of Goppa codes used in *Classic McEliece*.

Goppa codes

Definition 1 (Goppa code). *Let $g(x) = g_t x^t + \cdots + g_2 x^2 + g_1 x + g_0 \in \mathbb{F}_{q^m}[x]$, and let $L = \{\alpha_1, \alpha_2, ..., \alpha_n\} \subseteq \mathbb{F}_{q^m}$ such that, $g(\alpha_i) \neq 0$, for all $\alpha_i \in L$. Then the code defined by*

$$\mathcal{C} = \left\{ (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n) \in \mathbb{F}_q^n : \sum_{i=1}^{n} \frac{\boldsymbol{c}_i}{x - \alpha_i} \equiv 0 \mod g(x) \right\} \tag{2}$$

is called Goppa code with parameters L and $g(x)$, denoted by $\Gamma(L, g)$.

For each i, $\gcd(x - \alpha_i, g(x)) = 1$ since $g(\alpha_i) \neq 0$, the value of $(x - \alpha_i)^{-1}$ is computed into $\frac{\mathbb{F}_{q^m}[x]}{\langle g(x) \rangle}$.

Theorem 1. *[MS77] The multiplicative of $(x - \alpha_i)$ inverse exists in the quotient ring $\frac{\mathbb{F}_{q^m}[x]}{\langle g(x) \rangle}$; the value of $(x - \alpha_i)^{-1}$ in $\frac{\mathbb{F}_{q^m}[x]}{\langle g(x) \rangle}$ is $\left(\frac{g(\alpha_i) - g(x)}{x - \alpha_i}\right) g(\alpha_i)^{-1}$. A vector $\boldsymbol{c} \in \Gamma(L, g)$ if and only if $\sum_i c_i \left(\frac{g(\alpha_i) - g(x)}{x - \alpha_i}\right) g(\alpha_i)^{-1} \equiv 0 \mod g(x)$.*

Using this result, we can derive the following important corollary:

Corollary 1. *[MS77] For a Goppa code $\Gamma(L, g)$, the parity check matrix over $\mathbb{F}_{q^m}$ is*

$$\boldsymbol{H} = \begin{bmatrix} g(\alpha_1)^{-1} & g(\alpha_2)^{-1} & \dots & g(\alpha_n)^{-1} \\ \alpha_1 g(\alpha_1)^{-1} & \alpha_2 g(\alpha_2)^{-1} & \dots & \alpha_n g(\alpha_n)^{-1} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{t-1} g(\alpha_1)^{-1} & \alpha_2^{t-1} g(\alpha_2)^{-1} & \dots & \alpha_n^{t-1} g(\alpha_n)^{-1} \end{bmatrix}_{t \times n} . \tag{3}$$

If we consider the elements of $\mathbb{F}_{q^m}$ as vectors of length m over $\mathbb{F}_q$ by vector space isomorphism, we have a parity check matrix $\boldsymbol{H}$ for $\Gamma(L, g)$ over $\mathbb{F}_q$ to be a $mt \times n$ matrix, with at least t linearly independent columns over $\mathbb{F}_q$. Therefore, the Hamming distance of the Goppa code, $d(\Gamma(L, g)) \geq t + 1$. Since at most mt rows are linearly independent, $\text{Rank}(\boldsymbol{H}) \leq mt$ therefore, the dimension of the Goppa code is $k \geq n - mt$.

Definition 2 (Primitive polynomial). *An irreducible polynomial $p(z)$ of degree m over $\mathbb{F}_q$ is called a primitive polynomial if its roots form primitive elements of $\mathbb{F}_{q^m}$.*

Remark 1. The number of irreducible polynomials of degree t with coefficients in $\mathbb{F}_{q^m}$ is approximately $\frac{q^{mt}}{t}$.

2.2 Code-base cryptography

We briefly present the McEliece cryptosystem [McE78] and its variant, Niederreiter [Nie86], and the KEM *Classic McEliece* whose implementation on ARM Cortex-M4 is the target of our attack.

McEliece cryptosystem The McEliece cryptosystem was introduced by Robert J. McEliece in 1978. The basic idea of this cryptosystem is to use binary Goppa code ($q = 2$) with an efficient decoding algorithm that can correct up to t errors. The public key is a matrix $\boldsymbol{G} = \boldsymbol{S}\boldsymbol{G}'\boldsymbol{P}$ where $\boldsymbol{G}'$ is a generator matrix of the Goppa code, $\boldsymbol{S}$ (resp. $\boldsymbol{P}$) is a random $k \times k$ non-singular (resp. $n \times n$ permutation) matrix. The code length n, the code dimension k, and the Hamming weight of the error $wt(\boldsymbol{e}) = t$ are public parameters, but the Goppa code $\Gamma(L, g)$, matrices $\boldsymbol{S}$ and $\boldsymbol{P}$ are secrets. The encryption works by computing a code word for the plaintext $\boldsymbol{x}$ using the generator matrix $\boldsymbol{G}$ and by adding an error $\boldsymbol{e}$. The ciphertext $\tilde{\boldsymbol{c}}$ is therefore computed as $\tilde{\boldsymbol{c}} = \boldsymbol{x}\boldsymbol{G} \oplus \boldsymbol{e}$. The receiver corrects the error by applying the decoding algorithm with $\boldsymbol{G}'$ and recovers $\boldsymbol{x}$. The security of the system is based on the hardness of decoding a general linear code, a problem known to be $\mathcal{NP}-$complete [BMVT78].

Niederreiter cryptosystem This cryptosystem is a dual version of the McEliece cryptosystem. It was published by Harald Niederreiter in 1986. The main difference between the McEliece and the Niederreiter scheme is that the public key in Niederreiter's scheme is a parity check matrix instead of a generator matrix. In

the Niederreiter scheme, the ciphertext is a syndrome $\tilde{\boldsymbol{c}} = \boldsymbol{H}\boldsymbol{e}^T$ where the error vector $\boldsymbol{e}$ is the image of the plaintext $\boldsymbol{x}$ by an encoding function ϕ. Therefore, an efficient syndrome decoding algorithm is used for decryption. In what follows, we will focus on the instantiation of Niederreiter using binary Goppa codes as in *Classic McEliece* [Cho+20] (see also [BCS13] and [Cho17] for more details).

Classic McEliece *Classic McEliece* is a code-based a KEM introduced by Bernstein *et al.* [Ber+17]. It is composed of three algorithms (key generation, encapsulation and decapsulation) described below.

Key generation The key pair generation in *Classic McEliece* is described as follows:

1. Construct a parity check matrix $\boldsymbol{H}'$ for a Goppa code $\Gamma(L, g)$;
2. Transform $\boldsymbol{H}'$ into a $mt \times n$ binary matrix $\boldsymbol{H}$ by replacing each $\mathbb{F}_{2^m}$-entry by a m-bit column;
3. Compute the systematic form $[\boldsymbol{I}_{mt}|\boldsymbol{T}]$ of $\boldsymbol{H}$ and return $\{g(x), \alpha_1, \alpha_2, \ldots, \alpha_n\}$ as the secret key and $\boldsymbol{T}$ as the public key.

Encapsulation The session key K and its encapsulation are generated as follows:

1. Generate a random $\boldsymbol{e} \in \mathbb{F}_2^n$ with $wt(\boldsymbol{e}) = t$;
2. Compute the matrix $\boldsymbol{H} = [\boldsymbol{I}_{mt}|\boldsymbol{T}]$ by appending $\boldsymbol{T}$ to the identity matrix $\boldsymbol{I}_{mt}$ and then a vector $\tilde{\boldsymbol{c}}_0 = \boldsymbol{H}\boldsymbol{e}^T$;
3. Compute $\tilde{\boldsymbol{c}}_1 = \mathtt{H}(2|\boldsymbol{e})$ and generate ciphertext $\tilde{\boldsymbol{c}} = (\tilde{\boldsymbol{c}}_0|\tilde{\boldsymbol{c}}_1)$, $\mathtt{H}$ represents the hash function SHAKE256;
4. Compute a 256-bit session key $K = \mathtt{H}(1|\boldsymbol{e}|\tilde{\boldsymbol{c}})$.

Decapsulation Recovering the session key K' from a ciphertext $\tilde{\boldsymbol{c}}$ can be done as follows:

1. Split $\tilde{\boldsymbol{c}}$ as $(\tilde{\boldsymbol{c}}_0|\tilde{\boldsymbol{c}}_1)$, with $\tilde{\boldsymbol{c}}_0 \in \mathbb{F}_2^{mt}$ and $\tilde{\boldsymbol{c}}_1 \in \mathbb{F}_2^{256}$;
2. Use the underlying decoding algorithm for Niederreiter scheme to recover $\boldsymbol{e}$ such that $wt(\boldsymbol{e}) = t$ and $\tilde{\boldsymbol{c}}_0 = \boldsymbol{H}\boldsymbol{e}^T$;
3. Compute $\tilde{\boldsymbol{c}}_1' = \mathtt{H}(2|e)$, and checks if $\tilde{\boldsymbol{c}}_1' = \tilde{\boldsymbol{c}}_1$;
4. Compute the session key $K' = \mathtt{H}(1|e|\tilde{\boldsymbol{c}})$.

If there is no failure at any stage during the decapsulation process and $\tilde{\boldsymbol{c}}_1' = \tilde{\boldsymbol{c}}_1$, then surely the session key K' will be identical to K. In this scenario, the same session key is established. Table 1 shows the parameters of *Classic McEliece* proposed in[Cho+20] .

The main component of decapsulation in *Classic McEliece* is inspired by Chou *et al.* [Cho17]. This paper presents a fast constant-time implementation of the "McBits" proposed by Bernstein *et al.* [BCS13]. They use the FFT (Fast Fourier Transform) algorithms for root finding and syndrome computation, the Beneš network algorithm for secret permutation, and bitslicing for low-level operations.

Variant of KEM	m	n	t	Sec. level	Public key	Secret key	Ciphertext
`mceliece348864`	12	3488	64	1	261120	6492	128
`mceliece460896`	13	4608	96	3	524160	13608	188
`mceliece6688128`	13	6688	128	5	1044992	13932	240
`mceliece6960119`	13	6960	119	5	1047319	13948	226
`mceliece8192128`	13	8192	128	5	1357824	14120	240

Table 1: Parameter sets: sizes of public keys, secret keys and ciphertexts in bytes

3 Template Attack on *Classic McEliece*

Side-channel attacks are powerful tools for accessing secret information (passwords, secret keys, etc.) and pose a threat to cryptographic algorithm implementations. One of the most powerful techniques for evaluating side-channel information, template attack, was presented by Chari, Rao, and Rohatgi in [CRR02]. The general idea of template attacks is to rely on a multivariate model of side-channel traces to break secure implementations. In this section, we will first give a theoretical overview of template attacks, then we will show the information leakage related to the loading function of the coefficients of the Goppa polynomial during the decryption step of *Classic McEliece* and finally, we will present our template attack principle and results.

3.1 Template attacks

Multivariate-Gaussian model We consider the case where the electronic noise at each point of a power trace follows a normal distribution. This power consumption model does not take into account the correlation between neighboring points. To take into account this correlation between points, it is necessary to model a power trace t_r as a *multivariate normal distribution*. The multivariate normal distribution is a generalization of the normal distribution to higher dimensions. It can be described by a covariance matrix $\boldsymbol{C}$ and a mean vector μ. The Probability Density Function (PDF) of the multivariate normal distribution is given below.

$$PDF(t_r; (\mu, \boldsymbol{C})) = \frac{1}{\sqrt{(2\pi)^{\ell} \times \det(\boldsymbol{C})}} \exp\left(-\frac{1}{2}(t_r - \mu)^T \boldsymbol{C}^{-1}(t_r - \mu)\right) \quad (4)$$

where ℓ is the number of samples in the power trace t_r.

The covariance matrix $\boldsymbol{C}$ essentially characterizes the fluctuations in the power traces such as electronic noise. However, the multivariate normal distribution can also be used to characterize other components of power consumption. One of the problems with using the multivariate normal distribution is the cost of computing the covariance matrix $\boldsymbol{C}$ which grows quadratically with ℓ. Therefore, in practice, only small parts of the power traces are characterized.

General description Template attacks exploit that the power consumption also depends on the data that is being processed. In contrast to other types of power analysis attacks like Simple Power Analysis (SPA)[KJJ99] or Differential Power Analysis (DPA) [KJJ99]), template attacks consist of two phases. A *building phase*, in which the characterization of power consumption takes place, and a *matching phase*, in which the characterization is used to determine the secret information.

In template attack, the power traces can be characterized by a multivariate normal distribution, which is fully defined by a mean vector μ and covariance matrix $\boldsymbol{C}$. This couple $(\mu, \boldsymbol{C})$ is called a *template*. We can characterize the attacked device to determine the templates of some instruction sequences. In the *building phase*, we use another device of the same type as the one being attacked that we can fully control. On this device, we execute these instruction sequences with different data d_i and key k_j to record the resulting power consumption. Then, we aggregate the traces that correspond to a pair of (d_i, k_j), and estimate μ and $\boldsymbol{C}$. Thus, we obtain a template h for every pair (d_i, k_j).

$$h_{d_i,k_j} = (\mu, \boldsymbol{C})_{d_i,k_j}.$$

As mentioned above the size of the covariance matrix grows quadratically with the number of points in the trace. The inverse of $\boldsymbol{C}$, which is needed to compute the PDF, can be numerically problematic for large ℓ as shown by the authors in [EPW10]. We need to find a strategy to determine the points that contain the most information about the characterized instruction. This is called Points of Interest (POI) and we denote the number of POI by N_{POI}. There are several methods to find the POIs, such as the Sum of Differences [RO04] that we will use in this paper, the Sum Of Squared Differences (SOSD) [GLRP06], the Signal to Noise Ratio (SNR) [MOP08], the Principal Component Analysis (PCA) [Arc+06] etc.

Next, we use the characterization and a power trace of the attacked device to determine the key. In this *matching phase*, we evaluate the probability density function of the multivariate normal distribution with $(\mu, \boldsymbol{C}_{d_i,k_j})$ and the power trace of the attacked device. In other words, given a power trace t_r of the attacked device, and a template h_{d_i,k_j}, we compute the PDF:

$$PDF(t_r; (\mu, \boldsymbol{C}_{d_i,k_j})) = \frac{1}{\sqrt{(2\pi)^{N_{POI}} \times \det(\boldsymbol{C})}} \exp\left(-\frac{1}{2}(t_r - \mu)^T \boldsymbol{C}^{-1}(t_r - \mu)\right). \tag{5}$$

We do this for every template. As a result, we get the PDFs

$$PDF(t_r; (\mu, \boldsymbol{C})_{d_1,k_1}), \ldots, PDF(t_r; (\mu, \boldsymbol{C})_{d_D,k_K}).$$

The probabilities measure how well the templates match a given trace t_r and the highest value indicates the correct template. Since each template is associated with a key, we get an indication of the correct key. If all keys are equiprobable, the decision rule that minimizes the probability of a wrong decision is to decide for h_{d_i,k_j} if

$$PDF(t_r; h_{d_i,k_j}) > PDF(t_r; h_{d_i,k_v}) \forall v \neq j.$$

This is the Maximum Likelihood (ML) decision rule.

Templates with power models In addition to building templates for data and key pairs, there are other strategies. For example, if a device leaks the Hamming weight of the data, then moving the value 1 will result in the same power consumption as moving the value 2. Therefore, the template associated with value 1 will correspond to a trace in which value 1 is moved as well as the template associated with value 2. Thus, it is possible to build templates for values with different Hamming weights. Suppose we want to build templates for the output of a single-byte instruction sequence. We can simply build templates from 0 to 8 per byte. This is the approach we will take to build our templates for a decryption sequence in *Classic McEliece* implementation.

3.2 Measurement setup and leakage analysis

For our experiments, we used *Classic McEliece* variant `mceliece8192128` ($m = 13, n = 8192, t = 128$) where the coefficients of the Goppa polynomial are represented on m bits. But for the purpose of the implementation in [CC21], each coefficient will be represented on 2 bytes instead of 1 byte and 5 bits. We attack a software implementation of the decapsulation algorithm (loading function of the coefficients of the Goppa polynomial, `irr_load_32x`) running on an STM32F415RGT6 microcontroller [CC21]. The microcontroller features a 32-bit ARM Cortex-M4 core with 1 MB Flash memory and 192 kB SRAM.

The traces are acquired using the ChipWhisperer-Pro (CW1200) [OC14] which is an open-source embedded security analysis platform for recording and analyzing power consumption. All traces are acquired at a sample rate 105×10^6 samples per second (105 MS/s). Data acquisition is controlled by scripts running on PC. The ChipWhisperer measures power consumption during loading of the coefficients of the Goppa polynomial $g(x)$ of degree t. Once the acquisition is finished, the PC stores the measured trace on the hard disk. The measurement process is repeated depending on the desired number of traces. In traces, we can distinguish four patterns Fig. 1. These patterns are caused by the bitsliced representation used in the loading function `irr_load_32x`. Bitslicing is a simulation of hardware implementations in software. It is an implementation trick to speed up software implementations. It was introduced by Biham[Bih97] in 1997 for DES. The basic idea of bitslicing is to represent n-bit data as a bit in n distinct registers. On 32-bit registers, there are 31 unused bits in each register, which can be filled in the same fashion by taking 31 other independent n-bit data and putting each of their n bits in one of the registers. Bitwise operators on 32-bit (e.g. `AND, OR, XOR`) then act as 32 parallel operators. The bitslicing in this loading function `irr_load_32x` consists of transposing 32 16-bit coefficients into 16 different 32-bit registers for each round of the main loop. This loading function (Fig. 2) consists of the main loop to load the 128 coefficients in steps of 32 (hence the four patterns in Fig. 1) nested in two consecutive loops. These two loops each load 16 16-bit coefficients. Then, each of these 16 coefficients is transposed into 16 different 32-bit registers. Finally, a kind of assembly operation of

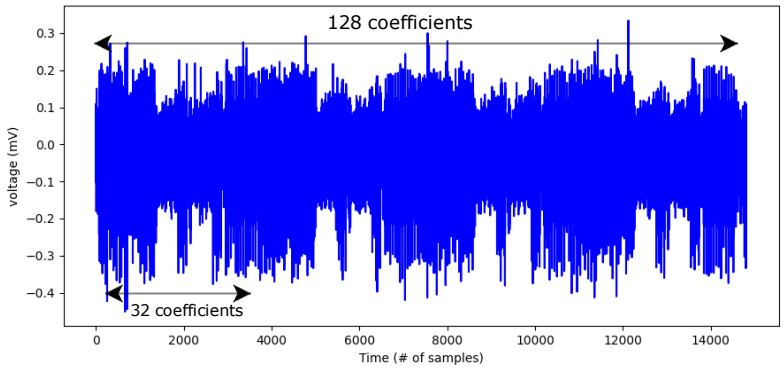

Fig. 1: Power consumption of the loading function of the Goppa polynomial.

these two groups is performed to have the 32 16-bit coefficients in 16 different 32-bit registers. The Fig. 3 shows the power consumption of this implementation strategy.

```
static inline void irr_load_32x(uint32_t out[][GFBITS], const
unsigned char * in, int len )
{
 int i, j;
 uint32_t mat[16];
 uint16_t *mat16 = (uint16_t*)&mat[0];
 for(i=0;i<len;i+=32) {
    for(j=0;j<16;j++) mat16[j] = load_gf(in + (i+j)*2);
    for(j=0;j<16;j++) mat16[16+j] = load_gf(in + (i+j+16)*2);
    transpose_16x16( mat16 , mat16 );
    transpose_16x16( (mat16+16), (mat16+16) );
    bs16_to_bs32( out[i>>5] , mat16 , mat16+16 , GFBITS );
  }
}
```

Fig. 2: Loading function of the coefficients of the Goppa polynomial $g(x)$ in [CC21]

We will now perform a power consumption analysis on the loading function `irr_load_32x`. The hypothesis is that small variations in power level may be observed in a trace based on the output of this function.
We analyze how the power consumption of the load function depends on the least significant bit (LSB) of 16-bit data with its transpose operation. If the LSB output produced by `irr_load_32x` is 1 then, in theory, the device under test should consume more power in comparison with a 0 value. If the hypothesis holds, we may exploit this fact to deduce the Hamming weights of the coefficients during

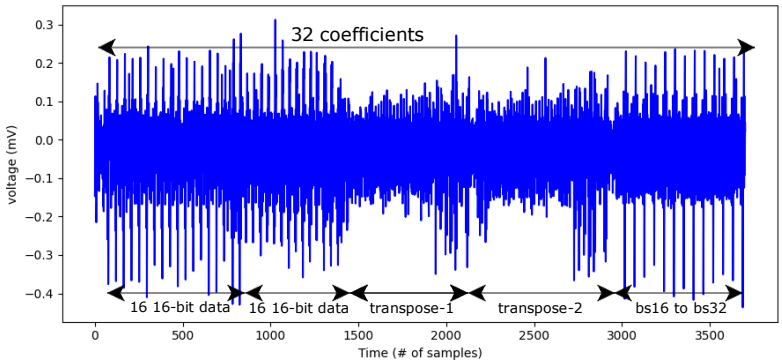

Fig. 3: Zoom on a pattern in power consumption of loading function of the coefficients of $g(x)$.

loading operations. It should be noted that this variation in power consumption is very small. The technique of differential power analysis used to determine the effect of the LSB on power consumption is the Difference Of Means (DOM). We measured the power consumption of the microcontroller when loading 10 000 random coefficients. We obtain 10 000 power traces, then sort them into two subsets (5 000 power traces for LSB=1 and 5 000 power traces for LSB=0) and calculate the average of each subset. The DOM between each subset will infer whether the proposed hypothesis is significant. In the case of a significant hypothesis, the DOM between the two subsets will highlight the change in power consumption when LSB of 0 is compared to LSB of 1. The average of each subset is calculated point by point. Thus, the difference of means can then be calculated by simply subtracting the points in the first subset from the points in the second subset. The result of this DOM is shown in Fig. 4.

3.3 Principle and results

We have two significant peaks easily discernible. These two peaks reveal the moments when the power consumption of the microcontroller depends on the LSB. Thus this loading function `irr_load_32x` leaks information. The transpose function also processes the same data as the loaded data and therefore the bitsliced representation according to this result also leads to information leakage. We now exploit this property in a power analysis attack to determine the Hamming weights of the Goppa polynomial coefficients $g(x)$. Recall that in *Classic McEliece*, the secret key consists of the Goppa polynomial and the support. Therefore, recovering the secret key corresponds to recovering $g(x)$ and L. It should also be noted that this experiment was done for the other 15 bits of the coefficient and the result is the same.

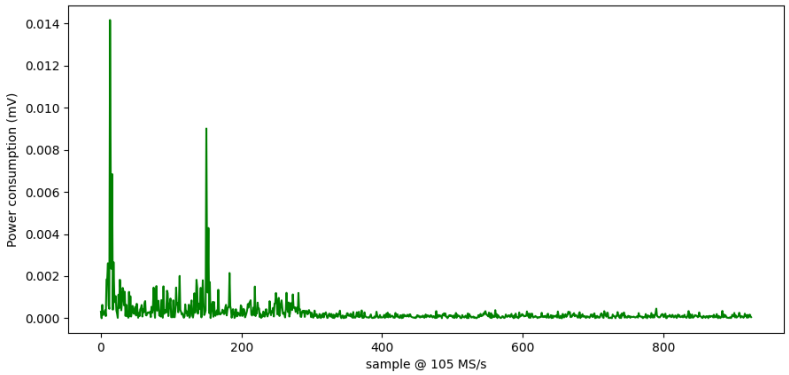

Fig. 4: Difference of means of power traces for LSB=1 and LSB=0.

In what follows, we will only use the loading of the first coefficient of $g(x)$ without the transposition for the template attack. In this section, we present the template attack on loading the first 16-bit coefficient of $g(x)$. We describe how to generate the templates from the random coefficient traces. Our template will attempt to recognize the Hamming weight of each coefficient at the output of the loading function `irr_load_32x`. The goal of our attack is to find the Hamming weights of all 128 coefficients of $g(x)$. This information will significantly reduce the complexity of the exhaustive search for the Goppa polynomial on $\mathbb{F}_{2^m}$. For our experiment, we used the same device for *building* and *matching phases*.
As we said at the end of Section 3.1, we will build templates in Hamming weights. Since each coefficient of $g(x)$ is represented on 16 bits in [CC21], we will build 17 templates for each coefficient. A key point with template attacks is that they require a large amount of data to make good templates. We recall that each output of this load function is unique, so there is only one output with a Hamming weight of 0 and one with a weight of 16. This means that using random inputs, there is only a $1/(256)^2$ probability of having a trace with a Hamming weight of 0 or 16.
In our experiment, we have efficiently used 350 000 traces, which allowed us to have a good distribution to build our templates. We then examine a trace and decide what its Hamming weight is. To set up the templates, we need to sort our template traces into 17 groups. The first group will consist of all traces that have a Hamming weight of 0. The next group has a Hamming weight of 1, etc., and the last group has a Hamming weight of 16. After sorting the traces by their Hamming weights, we look for the "average trace" for each weight. We create an array that contains 17 of these averages. We can always plot these averages to make sure that the average traces are correct. We will now use these average traces to find the POIs using the Sum of Differences method in Fig. 5. This method shows where these average traces have a lot of variances and the

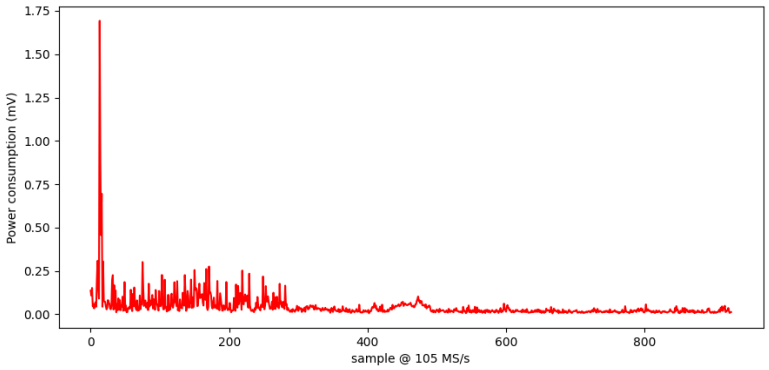

Fig. 5: Sum of Differences for 350 000 traces.

POIs are the highest peaks. In our case, we have one POI (sample 13). In the case where we had more peaks in the Sum of Differences, we cannot simply sort an array and choose the highest points. We have to make sure that the points are separated by some space. We can now construct our multivariate normal distribution for each Hamming weight. We need to write the $1 \times N_{POI}$ mean vector μ and the $N_{POI} \times N_{POI}$ covariance matrix $\boldsymbol{C}$ at this POI for each Hamming weight. In our case $N_{POI} = 1$, the mean is a scalar and the covariance matrix is reduced to the variance.

Our templates are ready, so we can use them to perform the *matching phase* now. For a decryption, using a random ciphertext, we recorded 17 power traces on the attacked device during the loading of the first coefficient of $g(x)$. These 17 power traces correspond to the loading of the coefficients with Hamming weights set from 0 to 16. We load our 17 template traces for each coefficient with a fixed Hamming weight and apply the *PDF* for each target trace to check which template fits best. We were able to find the Hamming weight of the first coefficient of the Goppa polynomial on the device attacked during decryption with a success rate of 99.86% after 1000 simulations. This *matching phase* with one target trace takes about 4 seconds on an 8-core processor running at 3.6 GHz. Thus we manage with our templates to find the Hamming weight of the first coefficient of $g(x)$ when it is loaded from memory. To find the Hamming weights of the remaining $t-1$ coefficients, it is necessary to construct $t-1$ groups of $m+1$ templates in Hamming weights. We proceed in the same way as for the first coefficient, except that the position of the POI(s) will change because it depends on the position of the coefficient during loading. The knowledge of the Hamming weights of the coefficients of $g(x)$ allowed us to improve the complexity of the exhaustive search for the Goppa polynomial on $\mathbb{F}_{2^m}$.

4 Complexity of the Goppa polynomial search

In this section, we have given the complexity of finding the secret Goppa polynomial in the original McEliece with the knowledge of Hamming weights of its coefficients. Indeed, the key recovery attack against McEliece using irreducible Goppa code consists of recovering the Goppa polynomial g and the support $L = \{\alpha_1, \alpha_2, ..., \alpha_n\} \in \mathbb{F}_{2^m}^n$. However, the best way to proceed is described as follows [LS01]:

1. *Step 1:* Find an monic irreducible polynomial g of degree t such that the Goppa code $\Gamma(L, g)$ is equivalent to the public code $\mathcal{C}$.
2. *Step 2:* Find the permutation by using the Support Splitting Attack (SSA) algorithm.

The cost of a such enumerative attack is given by $Cost = \lambda n^3 \mathcal{C}_{irr}$ where n is the code length and λ is a small constant depending on the implementation of the attack [LS01]. $\mathcal{C}_{irr}$ is the number of irreducible polynomials over $\mathbb{F}_{2^m}$ i.e the cardinal of search space. In the case of an extended code of Goppa, the value of $\mathcal{C}_{irr}$ is given by [LS01]

$$\mathcal{C}_{irr} \approx \frac{2^{m(t-3)}}{mt}. \tag{6}$$

With the knowledge of Hamming weights δ_j" g_j, and by assuming that irreducible monic polynomials over $\mathbb{F}_{2^m}$ have a uniform distribution, the cardinal of the search space is given by

$$\#\text{Search Space} = \frac{\prod_{j=0}^{t-1} \binom{m}{\delta_j}}{t}. \tag{7}$$

In fact, to the best of our knowledge, there is no specific algorithm to construct an irreducible polynomial by knowing the Hamming weights of its coefficients. Therefore, finding such polynomials corresponds in practice to search in the set of all monic polynomials with corresponding coefficients Hamming weights set. Note that, in the case that the Goppa polynomial is an irreducible binary polynomial that corresponds to a weak key in [LS01], with the knowledge of the Hamming weight of coefficients, the cardinal of the search space is equal to 1. Indeed, all coefficients with non-zero Hamming weight have their value equal to 1 thus, we can directly reconstruct the Goppa polynomial without searching (Table 2). While it was shown in [LS01] that the computation time to find this irreducible binary polynomial for an instance where $m = 10$, $n = 1024$, and $t = 50$ with their implementation should be 500 years. We can also imagine a scenario in which the user generates non-binary polynomials with fixed weights for all coefficients (to speed up the generation of the public key).

key level	#Search Space
Weak keys in [LS01]	$\frac{2^{m(t-3)}}{mt}$
Weak keys with knowledge of coefficients Hamming weight	1
Slightly less weak keys with coefficients Hamming weight equal to 1	$\frac{\binom{m}{1}^t}{t}$
Slightly less weak keys with coefficients Hamming weight equal to 2	$\frac{\binom{m}{2}^t}{t}$

Table 2: Size of the search space in the case of Weak keys

This technique only speeds up the generation of the public key offline but not the decryption process. We call this second case "slightly less weak keys". In the Table 2, we show the expression of the size of search space for two particular cases of "slightly less weak keys". It is sufficient to perform an exhaustive low-complexity search to recover the Goppa polynomial. Thus with the information on the Hamming weights of the coefficients of the Goppa polynomial, we have increased the set of weak keys proposed in [LS01].
For non-binary irreducible polynomials where the extension degree is larger than 8, each finite field element is implemented at least on 2 bytes. Thus, for suitable values $0 \leq i_j \leq \delta_j$, one can look for polynomials whose coefficients have Hamming weight i_j on the first byte and $\delta_j - i_j$ on the remaining $m-8$ bits. With this technique, the cardinality of the search space (7) becomes

$$\#\text{Search Space} = \frac{\prod_{j=0}^{t-1} \binom{8}{i_j}\binom{m-8}{\delta_j - i_j}}{t} \leq \frac{\binom{8}{4}^t \binom{m-8}{\lfloor \frac{m-8}{2} \rfloor}^t}{t}. \tag{8}$$

When extended code is implemented in the attack, this number should be divided by $2^m(2^m-1)m$. Indeed there are $2^m(2^m-1)m$ polynomial g' such that the extended codes of the Goppa codes $\Gamma(L,g')$ are equivalent to that of $\Gamma(L,g)$. Therefore, we can upper bound the number $\tilde{\mathcal{C}}_{irr}$ of irreducible polynomials with the knowledge of the Hamming weights of the polynomial coefficients by

$$\tilde{\mathcal{C}}_{irr} \leq \frac{\binom{8}{4}^t \binom{m-8}{\lfloor \frac{m-8}{2} \rfloor}^t}{2^m(2^m-1)mt}. \tag{9}$$

With our template attack on the implementation of *Classic McEliece*, we significantly reduced the cost of the exhaustive search of the Goppa polynomial in $\mathbb{F}_{2^m}$. As shown in Table 3, the knowledge of the Hamming weights of the Goppa polynomial coefficients allowed us for example to divide the size of the search space by 2^{441} for the variant of *Classic McEliece* with $m = 13$, $n = 8192$ and $t = 128$ from 2^{1615} to 2^{1174}. To date, we propose here the best complexity reduction for an exhaustive Goppa polynomial search.

5 Comparison with other key recovery attacks

We recall that the decryption in *Classic McEliece* is equivalent to the syndrome decoding of Goppa binary codes, including the steps of computing the syndrome

m	t	n	$\log_2(\mathcal{C}_{irr})$	$\log_2(\tilde{\mathcal{C}}_{irr})$				$\log_2(\mathcal{C}_{irr}/\tilde{\mathcal{C}}_{irr})$
				$\delta_j = 1$	$\delta_j = 2$	$\delta_j = 3$	$\delta_j = m/2$	
12	64	3488	725	158	274	338	534	191
13	96	4608	1199	251	425	521	871	328
13	128	6688	1615	347	578	706	1174	441
13	119	6960	1498	320	535	654	1089	409
13	128	8192	1615	347	578	706	1174	441

Table 3: Complexity to find Goppa polynomial with the knowledge of the Hamming weights of its coefficients from template attack against parameters of *Classic McEliece* [Cho+20]

polynomial and the error locator polynomial and that of its evaluation at points in $\mathbb{F}_{2^m}$. This polynomial evaluation over $\mathbb{F}_{2^m}$ is realized thanks to the implementation of the additive FFT after having calculated the error locator polynomial with the Berlekamp-Massey (BM) algorithm. Recently, Guo *et al.* [GJJ22] proposed a key-recovery side-channel attack on reference implementations (on FPGA and ARM Cortex-M4) [CC21; WSN18] of *Classic McEliece*. They design an attack algorithm in which they submit special ciphertexts to the decryption oracle that correspond to single error cases in the plaintexts. They exploited a leak in the additive FFT with a fixed input error before using a machine learning-based classification algorithm to determine the error locator polynomial. They choose a plaintext or error e of Hamming weight equal to 1 before encrypting it. Then, the profiled attack allows them to find the secret polynomial of the error locator among the 2^m possibilities and thus obtain an element of the support$L$. Finally, they designed new algorithms to recover the Goppa irreducible polynomial and then the full secret key.

Attack	Hamming weight	Target
Guo *et al.* [GJJ22]	1	The FFT additive and BM algorithm
Our attack	no constraints	Loading function of Goppa polynomial coefficients

Table 4: Profiled side-channel attacks on *Classic McEliece*

Countermeasures for the GJJ attack The main drawback of the key-recovery side-channel attack on *Classic McEliece* by Guo *et al.* [GJJ22] is the constraint on the Hamming weight of the plaintexts. This attack requires decrypting ciphertexts with Hamming weights of 1 to recover the secret in *Classic McEliece*. However, one can easily notice that with the systematic form of $\boldsymbol{H}$, we can detect the problem of bad ciphertexts with Hamming weights less than or equal to t. This is not the only step where this attack could be compromised. Recall that the error locator polynomial is obtained with the Berlekamp-Massey algorithm.

At this point, the error locator polynomial, via its degree, directly reflects any intentional error or misformated chipertext. Indeed, when this polynomial is of degree lower than t the decryption must stop and thus one can avoid the GJJ attack. All of these point out towards the fact that the GJJ attack to find the secret key in *Classic McEliece* is not realistic.

Positive points in our attack Our template attack to find the Hamming weights of the Goppa polynomial coefficients on *Classic McEliece* is realistic compared to the key-recovery side-channel attack of Guo *et al.* for the variant `mceliece8192128` as shown in the Table 4. First, we have no constraints on the Hamming weight of the ciphertexts and use fewer traces to recover the Hamming weight of Goppa polynomial coefficients. Secondly, we just follow the steps of the decryption execution in the implementation of *Classic McEliece* and we measure the trace of power consumption corresponding to the loading of the coefficients of the Goppa polynomial. Finally, our method has the particularity to be extended on other steps of decryption in *Classic McEliece* or other cryptosystems. Indeed, the loading of a vector of evaluation points is also performed just before the additive FFT for the evaluation of the error locator polynomial. If the setup is such that our attack can be reproduced we could gain the same information (Hamming weight) about the evaluation points. Also, notice that any decoding algorithm for binary Goppa codes has to use this/similar function for loading the Goppa polynomial and the points of evaluation. This information on the Hamming weights of the evaluation points, combined with our current results on the coefficients of the Goppa polynomial, will greatly improve our knowledge of the secret in *Classic McEliece*. We can also use this information on the evaluation points to apply the decoding method with a hint from [KM22] to directly recover the Goppa polynomial.

Common countermeasure fail against our attack We recall that our attack on the loading function of the Goppa polynomial coefficients is performed on the optimized reference implementation of *Classic McEliece* on ARM-Cortex M4 [CC21]. This reference implementation represents the side-channel attack target of several recent papers [Cay+21; Col+22b; GJJ22]. Shuffling is nowadays one of the most common and effective countermeasure techniques against most side-channel attacks [CMJ22]. However, in our attack, shuffling does not work because loading the coefficients in a random order will not affect their Hamming weights. In general, we have shown in this work that this loading function is not appropriate for secret variables in *Classic McEliece*. Moreover, our proposal is a first step towards much more powerful and potentially generic possible attacks. Indeed, the loading of a list of secret coefficients is performed at several places in the decryption of a code-based cryptographic scheme, and if we can isolate the exact moment when such functions are loaded then we could imagine applying our attack.

6 Conclusion

In this paper, we have presented a side-channel attack against the reference implementation of *Classic McEliece* on ARM Cortex-M4. The side channel here corresponds to loading the coefficients of the Goppa polynomial from memory during decryption. The *phase of matching* of our profiled attack is very fast (about 4 seconds) and allows us to find the Hamming weights of the coefficients of the Goppa polynomial that is an important part of the secret key of *Classic McEliece*. First, this result allowed us to directly find the Goppa polynomial in the case of weak keys with the method of Loidreau and Sendrier. In the case of "slightly less weak keys", we manage to find this polynomial with a low-complexity exhaustive search. Thus, we increase the set of weak keys compared to the method of Loiderau and Sendrier. Then, this information about the Hamming weights of the coefficients also allowed us to give the best complexity reduction for Goppa polynomial search on $\mathbb{F}_{2^m}$. Finally, we have shown that our attack is realistic (we only need a decryption of a random ciphertext) compared to other side-channel attacks on *Classic McEliece*. We also hope to apply it during in-memory loading of evaluation points to improve recent side-channel attack results on post-quantum cryptosystem implementations.

Acknowledgments

The author Jean Belo Klamti was supported by a grant of the Ripple Impact Fund/Silicon Valley Community Foundation (Grant 2018-188473).

References

[Arc+06] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. "Template attacks in principal subspaces". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2006, pp. 1–14.

[Ava+11] R. Avanzi, S. Hoerder, D. Page, and M. Tunstall. "Side-channel attacks on the McEliece and Niederreiter public-key cryptosystems". In: *Journal of Cryptograhic Engineering* 1.4 (2011), pp. 271–281.

[Bar+16] M. Bardet, J. Chaulet, V. Dragoi, A. Otmani, and J.-P. Tillich. "Cryptanalysis of the McEliece public key cryptosystem based on polar codes". In: *Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings 7*. Springer. 2016, pp. 118–143.

[BCS13] D. J. Bernstein, T. Chou, and P. Schwabe. "McBits: fast constant-time code-based cryptography". In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2013, pp. 250–272.

[Ber+17] D. J. Bernstein, T. Chou, T. Lange, I. von Maurich, R. Misoczki, R. Niederhagen, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, et al. "Classic McEliece: conservative code-based cryptography". In: *NIST submissions* (2017).

[Bih97] E. Biham. "A fast new DES implementation in software". In: *International Workshop on Fast Software Encryption*. Springer. 1997, pp. 260–272.

[BMVT78] E. Berlekamp, R. McEliece, and H. Van Tilborg. "On the inherent intractability of certain coding problems (corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386.

[Cay+21] 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.

[CB14] I. V. Chizhov and M. A. Borodin. "Effective attack on the McEliece cryptosystem based on Reed-Muller codes". In: *Discrete applied Math.* 24.5 (2014), pp. 273–280.

[CC21] M.-S. Chen and T. Chou. "Classic McEliece on the ARM Cortex-M4." In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 492.

[CD10] P.-L. Cayrel and P. Dusart. "McEliece/Niederreiter PKC: Sensitivity to Fault Injection". In: *International Conference on Future Information Technology*. Busan, South Korea, May 2010.

[Che+16] C. Chen, T. Eisenbarth, I. von Maurich, and R. Steinwandt. "Horizontal and Vertical Side Channel Analysis of a McEliece Cryptosystem". In: *IEEE Transactions on Information Forensics and Secuity*. 11.6 (2016), pp. 1093–1105.

[Cho17] T. Chou. "McBits revisited". In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2017, pp. 213–231.

[Cho+20] T. Chou, C. Cid, S. UiB, J. Gilcher, T. Lange, V. Maram, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, et al. "Classic McEliece: conservative code-based cryptography 10 October 2020". In: (2020).

[CMJ22] Z. Chen, Y. Ma, and J. Jing. "Low-Cost Shuffling Countermeasures Against Side-Channel Attacks for NTT-Based Post-Quantum Cryptography". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42.1 (2022), pp. 322–326.

[CML17] L Chen, D Moody, and Y. Liu. *NIST post-quantum cryptography standardization*. 2017.

[Col+22a] B. Colombier, V.-F. Dragoi, P.-L. Cayrel, and V. Grosso. "Physical Security of Code-based Cryptosystems based on the Syndrome Decoding Problem". In: *Cryptarchi Workshop*. Porquerolles, France, 2022.

[Col+22b] B. Colombier, V.-F. Drăgoi, P.-L. Cayrel, and V. Grosso. "Profiled Side-channel Attack on Cryptosystems based on the Binary Syndrome Decoding Problem". In: *IEEE Transactions on Information Forensics and Security* (2022).

[Col+23] B. Colombier, V. Grosso, P.-L. Cayrel, and V.-F. Drăgoi. *Horizontal Correlation Attack on Classic McEliece*. Cryptology ePrint Archive, Paper 2023/546. 2023.

[Cou+14] A. Couvreur, P. Gaborit, V. Gauthier-Umaña, A. Otmani, and J.-P. Tillich. "Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes". In: *Designs, Codes and Cryptography* 73.2 (2014), pp. 641–666.

[CRR02] S. Chari, J. R. Rao, and P. Rohatgi. "Template attacks". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2002, pp. 13–28.

[DJ92] D. Deutsch and R. Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558.

[EPW10] T. Eisenbarth, C. Paar, and B. Weghenkel. "Building a side channel based disassembler". In: *Transactions on computational science X*. Springer, 2010, pp. 78–99.

[Fey18] R. P. Feynman. "Simulating physics with computers". In: *Feynman and computation*. CRC Press, 2018, pp. 133–153.

[GI19] L. Gyongyosi and S. Imre. "A survey on quantum computing technology". In: *Computer Science Review* 31 (2019), pp. 51–71.

[GJJ22] Q. Guo, A. Johansson, and T. Johansson. "A Key-Recovery Side-Channel Attack on Classic McEliece Implementations". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2022), pp. 800–827.

[GLRP06] B. Gierlichs, K. Lemke-Rust, and C. Paar. "Templates vs. stochastic methods". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2006, pp. 15–29.

[Gro+23] V. Grosso, P. Cayrel, B. Colombier, and V. Dragoi. "Punctured Syndrome Decoding Problem - Efficient Side-Channel Attacks Against Classic McEliece". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Ed. by E. B. Kavun and M. Pehl. Vol. 13979. Lecture Notes in Computer Science. Munich, Germany: Springer, Apr. 2023, pp. 170–192.

[Gro96] L. K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[HMP10] S. Heyse, A. Moradi, and C. Paar. "Practical Power Analysis Attacks on Software Implementations of McEliece". In: *Third International Workshop on Post-Quantum Cryptography*. Ed. by N.

Sendrier. Vol. 6061. Lecture Notes in Computer Science. Darmstadt, Germany: Springer, May 2010, pp. 108–125.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. "Differential power analysis". In: *Annual international cryptology conference*. Springer. 1999, pp. 388–397.

[KM22] E. Kirshanova and A. May. "Decoding McEliece with a Hint – Secret Goppa Key Parts Reveal Everything". In: *Security and Cryptography for Networks*. Ed. by C. Galdi and S. Jarecki. Cham: Springer International Publishing, 2022, pp. 3–20.

[Lah+20] N. Lahr, R. Niederhagen, R. Petri, and S. Samardjiska. "Side Channel Information Set Decoding Using Iterative Chunking - Plaintext Recovery from the "Classic McEliece" Hardware Reference Implementation". In: *Annual International Conference on the Theory and Application of Cryptology and Information Security*. Ed. by S. Moriai and H. Wang. Vol. 12491. Lecture Notes in Computer Science. Daejeon, South Korea: Springer, Dec. 2020, pp. 881–910.

[Lar+21] M. V. Larsen, X. Guo, C. R. Breum, J. S. Neergaard-Nielsen, and U. L. Andersen. "Deterministic multi-mode gates on a scalable photonic quantum computing platform". In: *Nature Physics* 17.9 (2021), pp. 1018–1023.

[LS01] P. Loidreau and N. Sendrier. "Weak keys in the McEliece public-key cryptosystem". In: *IEEE Transactions on Information Theory* 47.3 (2001), pp. 1207–1211.

[McE78] R. J. McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116.

[Mis+13] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto. "MDPC-McEliece: New McEliece variants from Moderate Density Parity-Check codes". In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. 2013, pp. 2069–2073.

[Mol+11] H. G. Molter, M. Stöttinger, A. Shoufan, and F. Strenzke. "A simple power analysis attack on a McEliece cryptoprocessor". In: *Journal of Cryptographic Engineering* 1.1 (2011), pp. 29–36.

[MOP08] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media, 2008.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*. Vol. 16. Elsevier, 1977.

[Nie86] H. Niederreiter. "Knapsack-type cryptosystems and algebraic coding theory". In: *Prob. Contr. Inform. Theory* 15.2 (1986), pp. 157–166.

[OC14] C. O'Flynn and Z. D. Chen. "Chipwhisperer: An open-source platform for hardware embedded security research". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2014, pp. 243–260.

[OK15] A. Otmani and H. T. Kalachi. “Square Code Attack on a Modified Sidelnikov Cryptosystem”. In: *Codes, Cryptology, and Information Security*. Springer, 2015, pp. 173–183.

[Rav+22] P. Ravi, A. Chattopadhyay, J. P. D’Anvers, and A. Baksi. *Side-channel and Fault-injection attacks over Lattice-based Post-quantum Schemes (Kyber, Dilithium): Survey and New Results*. Cryptology ePrint Archive, Paper 2022/737. 2022.

[RO04] C. Rechberger and E. Oswald. “Practical template attacks”. In: *International Workshop on Information Security Applications*. Springer. 2004, pp. 440–456.

[Saa22] M.-J. O. Saarinen. “WiP: Applicability of ISO Standard Side-Channel Leakage Tests to NIST Post-Quantum Cryptography”. In: *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 2022, pp. 69–72.

[Sec+22] B. Seck, P.-L. Cayrel, I. Diop, V.-F. Dragoi, K. Couzon, B. Colombier, and V. Grosso. “Key-Recovery by Side-Channel Information on the Matrix-Vector Product in Code-Based Cryptosystems”. In: *International Conference on Information Security and Cryptology*. Seoul, South Korea, Nov. 2022.

[Sho94] P. W. Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[Sid94] V. M. Sidelnikov. “A public-key cryptosytem based on Reed-Muller codes”. In: *Discrete applied Math.* 4.3 (1994), pp. 191–207.

[SK14] S. R. Shrestha and Y.-S. Kim. “New McEliece cryptosystem based on polar codes as a candidate for post-quantum cryptography”. In: *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*. IEEE. 2014, pp. 368–372.

[TF19] S Takeda and A Furusawa. “Toward large-scale fault-tolerant universal photonic quantum computing”. In: *APL Photonics* 4.6 (2019), p. 060902.

[WSN18] W. Wang, J. Szefer, and R. Niederhagen. “FPGA-based Niederreiter cryptosystem using binary Goppa codes”. In: *International Conference on Post-Quantum Cryptography*. Springer. 2018, pp. 77–98.