FuLeeca: A Lee-based Signature Scheme

Stefan Ritterhoff, Georg Maringer, Sebastian Bitzer, Violetta Weger, Patrick Karl, Thomas Schamberger, Jonas Schupp, and Antonia Wachter-Zeh

Technical University of Munich, Germany

TUM School of Computation, Information and Technology[†]

{stefan.ritterhoff, georg.maringer, sebastian.bitzer, violetta.weger, patrick.karl, t.schamberger, jonas.schupp, antonia.wachter-zeh} @tum.de

Abstract. In this work we introduce a new code-based signature scheme, called FuLeeca, based on the NP-hard problem of finding low Lee-weight codewords. The scheme follows the Hash-and-Sign approach applied to quasi-cyclic codes of small Lee-weight density. Similar approaches in the Hamming metric have suffered statistical attacks, which reveal the small support of the secret basis. Using the Lee metric we are able to thwart such attacks. We use existing hardness results on the underlying problem and study adapted statistical attacks. We propose parameters for FuLeeca and compare them to the best known post-quantum signature schemes. This comparison reveals that FuLeeca is extremely competitive. For example, for NIST category I, i.e., 160 bit of classical security, we obtain an average signature size of 276 bytes and public key sizes of 389 bytes. This not only outperforms all known code-based signature schemes, but also the signature schemes Dilithium, Falcon and SPHINCS+ selected by NIST for standardization.

Keywords: Post-Quantum Cryptography · Signature scheme · Code-Based Cryptography · Lee metric

1 Introduction

In 1994, Shor [38] has introduced an algorithm, running in polynomial time on a capable quantum computer, to solve the integer factorization problem as well as the discrete logarithm problem. Most of the currently deployed public key cryptosystems and digital signature schemes are based on these problems and thus will be broken in the quantum-era. As a consequence, NIST announced a standardization competition in order to find viable solutions to replace the currently used cryptosystems with quantum-secure alternatives in 2016.

[†]The work of Georg Maringer was supported by the German Research Foundation, Deutsche Forschungsgemeinschaft (DFG), under Grant WA3907/4-1. Violetta Weger is supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 899987. Sebastian Bitzer acknowledges the financial support by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt.". Joint project 6G-life, project identification number: 16KISK002. This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 801434). The authors would like to thank Thomas Debris-Alazard and Sabine Pircher for useful discussions.

Since the standardization call several of the submitted cryptosystems have been broken or removed from the competition as the proposed parameters seemed inferior to other schemes. Recently, several cryptosystems have been selected for standardization, both for key encapsulation and digital signatures. The most competitive and already selected signature schemes in terms of parameter sizes (keys and signature sizes) are based on structured lattices, namely CRYSTALS-Dilithium [18] and Falcon [23]. If signature sizes or signing times are not a major concern, hash-based signatures like SPHINCS+[1] provide even smaller key sizes. However, schemes based on other hard problems featuring competitive key and signature sizes are required in case that a major breakthrough in attacking schemes based on structured lattices is achieved.

One possibility to build quantum-secure signature schemes is to rely on hard problems from coding theory that have been well examined over decades [9, 4, 3, 11]. In this work, we thus propose a code-based signature scheme in the Lee metric. While classical code-based cryptography considers vector spaces endowed with the Hamming metric, other metrics like the rank metric have attracted attention in the context of cryptography. This work marks the first Lee-based cryptographic primitive.

In general, there are two main methods to construct code-based signature schemes: the first one applies the Fiat-Shamir transform [22] to a code-based zero-knowledge identification scheme and the second one is called Hash-and-Sign approach [8]. The former approach usually suffers from huge signature sizes, due to large cheating probabilities within the identification scheme, and the latter features small signature sizes at the cost of larger public key sizes.

The signature scheme we present in this paper is based on the Hash-and-Sign approach, which has been introduced in 2001 by Courtois, Finiasz and Sendrier [15] (following the idea of [8]) and is often called the CFS scheme. This classical code-based Hash-and-Sign signature scheme is a direct adaption of the McEliece public-key encryption scheme. In fact, the rationale is to start with an algebraically structured secret code that comes with an efficient decoding algorithm. The public key is a disguised version of the secret code. To generate a signature, the message and a nonce is hashed until the digest results in a syndrome of a low-weight error vector. This approach has some potential drawbacks that have been exploited for attacks in the past: on the one hand, the public code might be distinguishable from a random code and thus leak information on the secret code[1]. On the other hand, the event that the hash of a message is a syndrome of a low weight codeword is highly unlikely and therefore this process has to be repeated many times. This causes the signing time of CFS to be impractically high. Additionally, as the public key is a disguised version of an algebraically structured code, the public key size of CFS tends to be rather large. The CFS scheme was the starting point for several Hash-and-Sign signature schemes, such as [2, 26, 14], which have not survived cryptanalysis [32, 34]. The code-based scheme WAVE [16] also follows the same blueprint but translated into the theoretical framework pioneered by Gentry, Peikert and Vaikuntanathan [25]. Additionally, it is based on the hardness of finding errors having large Hamming weights instead of small ones, thereby preventing all aforementioned attacks and so far no successful cryptanalysis has been mounted.

[1]See for example [21], where the CFS scheme using high rate Goppa codes has been attacked.

Code-based signature schemes based on quasi-cyclic structures with low Hamming density codes in the Hamming metric, e.g., [2, 33], are vulnerable to statistical key attacks, [36, 17]. These attacks have in common that they make use of the small support of the secret key. An attacker can recover the sparse secret key by observing the distribution of many signatures and comparing it to a random distribution. The use of the Lee metric thwarts such attacks, as even though the Lee weight of the secret basis is low, the number of non-zero entries is relatively large.

FuLeeca is therefore based on quasi-cyclic codes in the Lee metric. In a nutshell, the signature scheme works as follows: the secret key is a quasi-cyclic low Lee-weight generator matrix and the public key is its systematic form. Note that recovering the original low Lee-weight basis is as hard as the problem of finding low Lee-weight codewords, which has been proven to be NP-hard [40]. The binary hash output of the message $\boldsymbol{m}$ is mapped onto $\{\pm 1\}$ and is considered as the target vector $\boldsymbol{c}$ for the main step of the scheme: the signer uses the low Lee-weight basis to find a codeword, which is connected to the signature for $\boldsymbol{m}$, with two properties: firstly, the Lee weight should be low and secondly, the signum of the codeword should have many 1s, respectively -1s, in the same places as the target vector $\boldsymbol{c}$. This second property is used to bind the message to the signature, while the first property is essential to make the scheme secure.

For our chosen parameters targeting NIST security level I, the public key and signature sizes are only 389 bytes and 276 bytes, respectively. The sum of the public key and signature sizes is smaller than that of all signature schemes currently proposed within the NIST competition.

A multiple-use signature scheme should have an existential unforgeability under adaptive chosen message attacks (EUF-CMA) security proof. For code-based signatures constructed from a zero-knowledge identification scheme this property is assured through the number of rounds and the cheating probability. However, the EUF-CMA security proof is notoriously difficult for Hash-and-Sign approaches. To the best of our knowledge, WAVE [16] is the only known code-based Hash-and-Sign signature scheme that provides such a proof. Even though we do not provide a full security proof for FuLeeca, we consider attacks exploiting the leakage via published hash/signature pairs, and design our scheme integrating countermeasures for those attacks. Heuristically, we observe that our scheme does not leak any information via the standard attack vectors.

This paper is structured as follows: In Section 2 we introduce the notation that is used throughout this paper and recall the required coding-theoretic basics. In Section 3 we describe the proposed Lee-metric signature scheme FuLeeca. In Section 4 we analyze the security of the proposed scheme. We first consider the best known solver to find low Lee weight codewords, and secondly we provide heuristics for EUF-CMA security, which allow the signature scheme to be used multiple times. Finally, in Section 5 we analyze the performance of the scheme. We compare the key sizes, the signature size and the computation time for signing and verification to other post-quantum signature schemes. Section 6 concludes the paper.

2 Preliminaries

2.1 Notation

Throughout this work, we denote vectors in bold lowercase and matrices in bold uppercase letters. We refer to the i-th element of the vector $\boldsymbol{v}$ by v_i and similarly, to the j-th row of a matrix $\boldsymbol{A}$ by $\boldsymbol{a}_j$ and we denote the element in the j-th row and k-th column by $a_{j,k}$. The identity matrix of size n is denoted by $\mathbf{I}_n$. We denote by uppercase letters sets and for a set $S \subset \{1, \dots, n\}$, we denote by $|S|$ the cardinality and by $S^C = \{1, \dots, n\} \setminus S$ the complement.

The sampling of an element a from the uniform distribution over a set $\mathcal{K}$ is denoted by $a \overset{\$}{\leftarrow} \mathcal{K}$. While the sampling of an element a according to a distribution χ is given by $a \overset{\$}{\leftarrow} \chi$ and by a slight abuse of notation we denote sampling of a vector $\boldsymbol{v}$ independently and identically distributed (i.i.d.) from χ by $\boldsymbol{v} \overset{\$}{\leftarrow} \chi$. We denote the entropy of a random variable X by $H(X)$.

Throughout this paper, we denote by $\mathbb{F}_p$ the finite field of order p, where p is a prime. We often choose to represent this prime field as $\{-\frac{p-1}{2}, \dots, 0, \dots, \frac{p-1}{2}\}$, which we call the symmetric representation. For a set $S \subset \{1, \dots, n\}$ of size s and matrix $\boldsymbol{A} \in \mathbb{F}_p^{k \times n}$, we denote by $\boldsymbol{A}_S$ the $k \times s$ matrix formed by the columns of $\boldsymbol{A}$ indexed by S, similarly for a vector $\boldsymbol{x} \in \mathbb{F}_p^n$, we denote by $\boldsymbol{x}_S$ the vector of length s formed by the entries of $\boldsymbol{x}$ indexed by S.

2.2 Lee-metric Codes

An $[n, k]$ *linear code* $\mathcal{C}$ is a k-dimensional linear subspace of $\mathbb{F}_p^n$. An $[n, k]$ linear code can be compactly represented either through a *generator matrix* $\boldsymbol{G} \in \mathbb{F}_p^{k \times n}$, which has the code as its image or through a *parity-check matrix* $\boldsymbol{H} \in \mathbb{F}_p^{(n-k) \times n}$ having the code as its kernel. The elements of a code are called *codewords* and for any $\boldsymbol{x} \in \mathbb{F}_p^n$, we call $\boldsymbol{s} = \boldsymbol{x}\boldsymbol{H}^\top$ the *syndrome* of $\boldsymbol{x}$. The *rate* of an $[n, k]$ code is $R = \frac{k}{n}$.

For an $[n, k]$ linear code $\mathcal{C}$ and a set $I \subset \{1, \dots, n\}$, we denote by $\mathcal{C}_I$ the set of restrictions on codewords restricted to the coordinates specified in I. We say that $I \subset \{1, \dots, n\}$ of size k is an *information set*, if $|\mathcal{C}_I| = |\mathcal{C}|$. As a consequence, we have that for a generator matrix $\boldsymbol{G}$, respectively a parity-check matrix $\boldsymbol{H}$ of the code, $\boldsymbol{G}_I$ and $\boldsymbol{H}_{I^C}$ are invertible. We say that a generator matrix $\boldsymbol{G}$, respectively a parity-check matrix $\boldsymbol{H}$ are in systematic form (with respect to I), if $\boldsymbol{G}_I = \mathbf{I}_k$, respectively $\boldsymbol{H}_{I^c} = \mathbf{I}_{n-k}$.

Classically, in coding theory the vector space $\mathbb{F}_p$ is endowed with the Hamming metric, where the *Hamming weight* of a vector $\boldsymbol{v}$, denoted by $\mathrm{wt}_H(\boldsymbol{v})$, is given by the number of non-zero entries of $\boldsymbol{v}$. However, for this paper we are interested in a different metric, called the Lee metric.

The *Lee weight* of an element $a \in \mathbb{F}_p$ is defined as

$$\mathrm{wt}_L(a) := \min\{a, p - a\}, \tag{1}$$

where the representation of a is chosen to be in $\{0, \dots, p - 1\}$. In fact, one can think of the Lee weight as the L_1-norm modulo p. Clearly, the Lee weight of an element can be at most $(p - 1)/2$, thus we will denote this value by M. For a vector $\boldsymbol{v} \in \mathbb{F}_p^n$ its Lee weight is defined as the sum of the Lee weights of its elements, i.e.,

$$\mathrm{wt}_L(\boldsymbol{v}) := \sum_{i=1}^{n} \mathrm{wt}_L(v_i). \tag{2}$$

Notice that a vector of relatively large Lee weight does not have to be of large Hamming weight and vice versa (the Lee weight and Hamming weight coincide only for $p = 2$ and $p = 3$). In fact, we have that $\mathrm{wt}_H(\boldsymbol{v}) \leq \mathrm{wt}_L(\boldsymbol{v}) \leq M\mathrm{wt}_H(\boldsymbol{v})$ and the average Lee weight of the vectors in $\mathbb{F}_p^n$ is given by $nM/2$. For our scheme the aforementioned connections between Lee and Hamming weight are essential as they enable us to use generator matrices of low Lee weight that nevertheless have substantial Hamming weight.

The Lee weight induces the *Lee distance*, which we define by $d_L(\boldsymbol{x}, \boldsymbol{y}) := \mathrm{wt}_L(\boldsymbol{x}-\boldsymbol{y})$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_p^n$. For a linear code $\mathcal{C}$ we define the *minimum Lee distance* to be

$$d_L(\mathcal{C}) = \min\{\mathrm{wt}_L(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathcal{C}, \boldsymbol{c} \neq 0\}.$$

We denote by δ the relative minimum Lee distance, that is $\delta = \frac{d_L(\mathcal{C})}{nM}$. Let us denote by $V_L(p, n, r)$ the *Lee sphere* of radius t

$$V_L(p, n, t) := \{\boldsymbol{x} \in \mathbb{F}_p^n \mid \mathrm{wt}_L(\boldsymbol{x}) = t\},$$

and by

$$F_L(p, T) = \lim_{n\to\infty} \frac{1}{n} \log_p(|V_L(p, n, TnM)|)$$

its asymptotic size. The exact formulas for the size of $V_L(p, n, t)$ and $F_L(p, T)$ can be found in [40].

Let us denote by $A(n, \delta)$ the maximal size of a code in $\mathbb{F}_p^n$ of minimum Lee distance δMn and by

$$R(\delta) = \limsup_{n\to\infty} \frac{1}{n} \log_p(A(n, \delta)).$$

The Gilbert-Varshamov bound in the Lee-metric then states: $R(\delta) \geq 1 - F_L(p, \delta)$. In [12] it was shown that random Lee-metric codes attain with high probability the Lee-metric Gilbert-Varshamov (GV) bound, i.e., a random code has with high probability a relative minimum Lee distance δ such that $R(\delta) = 1 - F_L(p, \delta)$. If $\mathcal{C} \in \mathbb{F}_p^n$ is a random code of dimension k, we can also compute the expected number of codewords of a given Lee weight w as $|V_L(p, n, w)|p^{k-n}$.

2.3 Basic Cryptographic Tools

We denote the security parameter assuming a classical attacker by λ and the security parameter for an attacker having access to a capable quantum computer by λ'. We use standard definitions of probabilistic polynomial time algorithms. We denote by "Hash" a Hash function in the perfect random oracle model.

In a digital signature scheme we have two parties: *the signer* and the *verifier*. The signer randomly samples a secret key sk and computes and publishes the connected public key pk. Given a message $\boldsymbol{m}$, the signer then uses the secret key sk to compute a signature $\boldsymbol{v}$. The signer then sends $(\boldsymbol{m}, \boldsymbol{v})$ to the verifier. The verifier checks the validity of the signature $\boldsymbol{v}$ for the message $\boldsymbol{m}$ under the constraints imposed by the scheme using the public key. An adversary might try to construct a valid signature, either using just the knowledge of the public key, or after having observed several signatures corresponding to different messages. The adversary is only allowed to succeed with negligible probability, e.g., $< 2^{-\lambda}$.

3 System Description

In this section, we describe the digital signature scheme FuLeeca.

For our scheme we represent the elements of $\mathbb{F}_p$ within

$$\left\{-\frac{p-1}{2}, \dots, 0, \dots, \frac{p-1}{2}\right\}$$

for $p > 3$ prime and $n \in \mathbb{N}$ even. As usual, we write M for the maximal Lee weight in $\mathbb{F}_p$, that is $M = \frac{p-1}{2}$. We define a function $\operatorname{sgn}(x)$, that gives us the sign of an element in $\mathbb{F}_p$.

Definition 1 (Signum). *For* $x \in \mathbb{F}_p = \left\{-\frac{p-1}{2}, \dots, 0, \dots, \frac{p-1}{2}\right\}$ *let*

$$\operatorname{sgn}(x) = \begin{cases} 0 & \textit{if } x = 0, \\ 1 & \textit{if } x > 0, \\ -1 & \textit{if } x < 0. \end{cases}$$

For the symmetric representation of $\mathbb{F}_p$ this corresponds to the common signum function. Furthermore, we define a matching function $\operatorname{mt}(\mathbf{x}, \mathbf{y})$ that compares $\mathbf{x}$ and $\mathbf{y}$ and counts the number of symbols that hold the same sign.

Definition 2 (Sign Matches). *Let* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_p^n$ *and consider the number of matches in their sign such that*

$$\operatorname{mt}(\boldsymbol{x}, \boldsymbol{y}) = |\{i \in \{1, \dots, n\} \mid \operatorname{sgn}(x_i) = \operatorname{sgn}(y_i), x_i \neq 0, y_i \neq 0\}|.$$

We are interested in a close estimate of the probability of an attacker being able to reuse any of the previously published signatures. For that, we introduce a function calculating the probability that a vector and a uniformly random hash digest (in $\{\pm 1\}^n$) have at least $\hat{\mu}$ sign matches. When talking about the security of the signature scheme, we will usually consider the negative $\log_2$ of this probability.

Definition 3 (Logarithmic Matching Probability (LMP)). *For a fixed* $\boldsymbol{v} \in \mathbb{F}_p^n$ *and* $\boldsymbol{c} \xleftarrow{\$} \{\pm 1\}^n$, *the probability of* $\boldsymbol{c}$ *to have* $\mu := \operatorname{mt}(\boldsymbol{c}, \boldsymbol{v})$ *sign matches with* $\boldsymbol{v}$ *is*

$$B(\mu, \operatorname{wt}_H(\boldsymbol{v}), 1/2),$$

where $B(k, n, p)$ *is the binomial distribution defined as*

$$B(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k} \ .$$

To ease notation, we write $\operatorname{LMP}(\boldsymbol{v}, \boldsymbol{c}) = -\log_2(B(\mu, \operatorname{wt}_H(\boldsymbol{v}), 1/2))$.

Remark 4. The $\operatorname{LMP}(\boldsymbol{v}, \boldsymbol{c})$ is closely related to the number of sign matches between $\boldsymbol{v}$ and $\boldsymbol{c}$. We use the LMP to justify that message and signature are bound together. The difference compared to just counting the sign matches between $\boldsymbol{v}$ and $\boldsymbol{c}$ is that the LMP also takes into account that the Hamming weight of $\boldsymbol{v}$, i.e. the necessary amount of sign matches depends on $\operatorname{wt}_H(\boldsymbol{v})$. Specifically, for a set of published codewords, the problem of finding a message hash $\boldsymbol{c} = \operatorname{Hash}(\boldsymbol{m}||\text{nonce})$ having the required number of matches when compared with any of the codewords should be equivalent to the problem of finding a (second-)preimage for some hash function producing outputs of $\mathrm{LMP}(\boldsymbol{v}, \boldsymbol{c})$ bits. We consider this problem to be hard if the deployed hash function is secure. Throughout this work we assume access to a perfect hash function (Random Oracle).

In [5], the authors computed the marginal distribution of entries where vectors are uniformly distributed on a Lee sphere $V_L(p, n, w)$. As n tends to infinity we have the following result on the distribution of the elements in $\boldsymbol{x} \in \mathbb{F}_p^n$.

Lemma 5 ([5, Lemma 1]). *Given a vector $\boldsymbol{x} \in V_L(p, n, w)$. Then, for any $x \in \mathbb{F}_p$, the probability that some entry $\boldsymbol{x}_i$ is equal to x, is independently of i given by*

$$p_w(x) = \frac{1}{Z(\beta)} \exp(-\beta \operatorname{wt}_L(x)),$$

where Z denotes the normalization constant and β is the unique solution to $w = \sum_{i=0}^{p-1} \operatorname{wt}_L(i) p_i(x)$.

Definition 6 (Typical Lee Set). *For a fixed weight w, let $p_w(x)$ be the probability from Lemma 5 of the element $x \in \mathbb{F}_p$. Then we define the typical Lee set as*

$$T(p, n, w) = \left\{ \boldsymbol{x} \in \mathbb{F}_p^n \mid \boldsymbol{x}_i = x \text{ for } \lfloor p_w(x) n \rceil \text{ coordinates } i \in \{1, \ldots, n\} \right\},$$

i.e., the set of vectors, for which the element x occurs $\lfloor p_w(x) n \rceil$ times.

3.1 Key Generation

The key generation of our signature scheme is presented in Algorithm 1. The basic idea to generate the secret key $\boldsymbol{G}_{sec}$ is to sample two cyclic matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{F}_p^{n/2 \times n/2}$ of low Lee weight w_{key}, where $\boldsymbol{A}$ has to fulfill the extra property of being an invertible matrix. Note that this property is satisfied for random matrices with large probability. The public key is obtained by computing the row reduced echelon form of $\boldsymbol{G}_{sec}$, referred to as $\boldsymbol{G}_{sys}$. The public key is then formed by the non-trivial part of $\boldsymbol{G}_{sys}$, which we denote by $\boldsymbol{T}$.

Algorithm 1: Key Generation

```
Input: public parameters for prime $p$, code length $n$, security level $\lambda$, Lee weight $w_{\mathrm{key}}$
Output: public key $\boldsymbol{T}$, private key $\boldsymbol{G}_{\mathrm{sec}}$
1 $\boldsymbol{a}, \boldsymbol{b} \xleftarrow{\$} T(p, n/2, w_{key})$.
2 Construct cyclic matrix $\boldsymbol{A} \in \mathbb{F}_p^{n/2 \times n/2}$ from all shifts of $\boldsymbol{a}$. $\boldsymbol{A}$ needs to be invertible. If it's not, resample
    $\boldsymbol{a}$ according to Step 1
3 Construct $\boldsymbol{B} \in \mathbb{F}_p^{n/2 \times n/2}$ from all shifts of $\boldsymbol{b}$.
4 $\boldsymbol{G}_{\mathrm{sec}} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \end{pmatrix} \in \mathbb{F}_p^{n/2 \times n}$.
5 $\boldsymbol{G}_{\mathrm{sys}} = \begin{pmatrix} \mathbf{I}_{\mathbf{n/2}} & \boldsymbol{T} \end{pmatrix}$ of $\boldsymbol{G}_{\mathrm{sec}}$ with $\boldsymbol{T} = \boldsymbol{A}^{-1}\boldsymbol{B}$.
```

Note that $|T(p, n/2, w_{key})|^2$ corresponds to the cardinality of our key space. In order to prevent brute force attacks this cardinality needs to be larger than 2^λ.

3.2 Signature Generation

Most of the Hash-and-Sign schemes require the Hash of a message to be a syndrome for a public parity-check matrix. In this Hash-and-Sign algorithm, however, we proceed differently and use the generator matrix to generate signatures which are low Lee-weight codewords. The connection to the Hash of the message vector is established through a number of sign matches that is a lot larger than the expected number of matches for a unrelated Hash.

The signature generation takes as input the message $\boldsymbol{m}$ to be signed and uses the private key $\boldsymbol{G}_{sec}$ to generate the signature $\boldsymbol{y}$. To do so, the algorithm utilizes the short basis of the code, namely the rows of $\boldsymbol{G}_{sec}$, to find a codeword $\boldsymbol{v} = [\boldsymbol{y}, \boldsymbol{yT}]$ of Lee weight slightly less than w_{sig}. Without having access to a short basis (the private key) it is computationally hard to find codewords in the desired Lee weight range. This property is important to make the scheme secure. Loosely speaking, the property that enough signs between the codeword $\boldsymbol{v}$ and Hash($\boldsymbol{m}$||nonce) are matching establishes the connection between the signature and the message. Furthermore, assuming the Hash function to be a random oracle, changing the nonce if a signing attempt does not work, guarantees that any message can be signed successfully.

Step 5 assures that the Hash of the message, i.e., the vector $\boldsymbol{c}$, is in $\{\pm 1\}^n$ making its signs comparable with the signs of vectors in $\mathbb{F}_p^n$. In Step 8, we are checking how many matches the row $\boldsymbol{g}_i$ has with the target vector $\boldsymbol{c}$. We take into account how many of the signs of $\boldsymbol{c}$ and $\boldsymbol{g}_i$ are matching in Step 9. We do this by setting the magnitude in the corresponding position of the information vector according to the number of matches and the scaling factor s. Thus, if the row has has a number matches when compared with the target $\boldsymbol{c}$ that is noticeably above average, we add this row multiple times. Similarly, if the number of matches is far below average, we subtract the same row multiple times.

Steps 10-26, which we refer to as the *Concentrating* procedure, are necessary to ensure that the signatures vary as little as possible in Lee weight and sign matches. To have signatures with much lower Lee weight than other signatures or varying LMP is undesirable, as this might leak information on the secret key. Thus, the iterative approach in lines 10-26 is used to obtain signatures with an almost equal matching probability (LMP) and almost constant Lee weight.

Note that, since we greedily minimize the absolute distance to the targeted LMP, the algorithm will choose to add or subtract rows to increase or decrease the number of sign matches as necessary. Interestingly, we don't have to control the (relative) Lee weight of the code word explicitly, since both adding and subtracting rows will slowly increase the weight with high probability (assuming we do not undo earlier decisions). Once we would exceed the targeted weight by adding or subtracting an additional row, we stop. If additionally, the observed LMP exceeds the threshold specified for verification, we return the systematic portion (just the first half) of the produced codeword together with the nonce. Else we discard the failed attempt, sample a new nonce and repeat the procedure from the beginning.

In Step 20 and 23 we ensure that a row of the generator matrix that has been added throughout the iterative procedure will not be subtracted in any of the subsequent steps. Similarly, this also holds for rows that have been subtracted. This is necessary to avoid falling into infinite loops.

Algorithm 2: Signing

```
Input: secret key $\boldsymbol{G}_{sec}$ with rows $\boldsymbol{g}_i$, secret matrix $\boldsymbol{A}$, message $\boldsymbol{m}$, thresholds $\varepsilon$, signature weight $w_{sig}$,
       scaling factor $s \in \mathbb{R}$, security level $\lambda$.
Output: signature = (nonce, $\boldsymbol{y}$).
1  Set $\boldsymbol{G} = \begin{pmatrix} \boldsymbol{G}_{sec} \\ -\boldsymbol{G}_{sec} \end{pmatrix}$ with rows $\boldsymbol{g}'_i$
2  repeat
3      nonce $\xleftarrow{\$} \{0,1\}^{\lambda+64}$;                                  // Simple signing starts
4      $\boldsymbol{c} \leftarrow$ Hash($\boldsymbol{m}$ || nonce)
5      $c_i \leftarrow (-1)^{c_i} \quad \forall i$
6      $\boldsymbol{x} \leftarrow (0,\dots,0)$
7      for $i \leftarrow 1$ to $n/2$ do
8          $x_{mt} = \mathrm{mt}(\boldsymbol{g}_i, \boldsymbol{c}) - \frac{\mathrm{wt}_H(\boldsymbol{g}_i)}{2}$
9          $\boldsymbol{x}_i = \lfloor x_{mt} s \rfloor$ ;                                  // Simple signing ends
       end
10     $\mathcal{A} \leftarrow \{1,\dots,n\}$ ;                                        // Allowed row index set
       repeat
11         $\boldsymbol{\nu} \leftarrow (0,\dots,0)$;                                   // Concentrating starts
12         $\boldsymbol{w} \leftarrow (0,\dots,0)$
13         for $i \in \{1,\dots,n\}$ do
14             $\boldsymbol{\nu}_i \leftarrow |\mathrm{LMP}(\boldsymbol{x}\boldsymbol{G}_{sec} + \boldsymbol{g}'_i, \boldsymbol{c}) - (\lambda + 64 + \varepsilon)|$ ;    // Difference to target LMP
15             $\boldsymbol{w}_i \leftarrow \mathrm{wt}_L(\boldsymbol{x}\boldsymbol{G}_{sec} + \boldsymbol{g}'_i)$
           end
16         $i \leftarrow \mathrm{argmin}_{\mathcal{A}} \boldsymbol{\nu}_\ell$
17         if $\boldsymbol{w}_i \leq w_{sig}$ then
18             if $i \leq \frac{n}{2}$ then
19                 $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + 1$
20                 $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i + n/2\}$
21             else
22                 $\boldsymbol{x}_{i-n/2} \leftarrow \boldsymbol{x}_{i-n/2} - 1$
23                 $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i - n/2\}$
24         else if $\mathrm{LMP}(\boldsymbol{x}\boldsymbol{G}_{sec}, \boldsymbol{c}) \geq \lambda + 64$ then
25             return (nonce, $\boldsymbol{y} \leftarrow \boldsymbol{x}\boldsymbol{A}$)
26         else
27             go to Step 3 ;                                                     // Concentrating ends
       end
   end
```

We also remark that the codeword corresponding to the signature $\boldsymbol{xG}_{sec}$ will never be close to any of the rows in the secret basis $\boldsymbol{G}_{sec}$. This is due to the number of sign matches between these rows and the target vector $\boldsymbol{c}$ being far too low. In fact, only after having added several rows we can ensure that the signs of $\boldsymbol{xG}_{sec}$ are close to the target LMP. Furthermore, we exploit the fact that adding rows of the secret generator matrix increases the Lee weight with high probability, thereby we are able to generate signatures marginally below the target Lee weight.

3.3 Verification

The verification process is quite simple and given in Algorithm 3. First, the verifier checks that $\boldsymbol{v}$ is indeed a codeword of the public code. This is ensured by computing $\boldsymbol{v}$ as $[\boldsymbol{y}\ \boldsymbol{yT}]$. Then, the verifier checks that the codeword $\boldsymbol{v}$ has a Lee weight of at most w_{sig}. Finally, one checks whether a sufficient amount of the signs of the non-zero elements of the codeword $\boldsymbol{v}$ match Hash($\boldsymbol{m}$||nonce), i.e., $\mathrm{LMP}(\boldsymbol{v}, \boldsymbol{c}) \geq \lambda + 64$.

Algorithm 3: Verification

```
Input: signature $\boldsymbol{y}$, message $\boldsymbol{m}$, nonce, public key $\boldsymbol{T}$, Lee weight $w_{\mathrm{sig}}$.
Output: Accept or Reject
1 $\boldsymbol{c} \leftarrow \mathrm{Hash}(\boldsymbol{m}\ ||\ \mathrm{nonce})$
2 $c_i \leftarrow (-1)^{c_i} \quad \forall i$
3 $\boldsymbol{v} = [\boldsymbol{y}\ \boldsymbol{yT}]$.
4 Accept if the following two conditions are satisfied:
  (a) $\mathrm{wt}_L(\boldsymbol{v}) \leq w_{\mathrm{sig}}$,
  (b) $\mathrm{LMP}(\boldsymbol{v}, \boldsymbol{c}) \geq \lambda + 64$.

  Otherwise, Reject.
```

3.4 Signature Compression

Notice that the verifier can compute $\boldsymbol{h} = (-1)^{\mathrm{Hash}(\boldsymbol{m}||\mathrm{nonce})}$ and thus also $\boldsymbol{y}' = \boldsymbol{y} \star \boldsymbol{h}$, where $\star$ is the component-wise multiplication. Since $\boldsymbol{y} = \boldsymbol{y}' \star \boldsymbol{h}$, it is enough to send $\boldsymbol{y}'$, and the verifier is able to recover $\boldsymbol{v}$. This reduces the entropy in the sent vector since most of the signs in $\boldsymbol{h}$ match the signs of the signature.

Note that $\boldsymbol{y}'$ is heavily concentrated rather than uniform over $\mathbb{F}_p$ (this is similar to Falcon [23]). We make use of the well known fact that it is possible to losslessly compress $\boldsymbol{y}'$ close to its entropy [20], thereby reducing the signature size of the scheme.

4 Security Analysis

In this section, we assess the security of the proposed system. The analysis consists of three parts: we begin by considering the generic solvers for the low Lee weight codeword finding problem. The second part describes known attacks and our countermeasures. Finally, we take all those attacks into account we show that the presented parameters achieve the security levels required by NIST.

4.1 The underlying hard problem and generic solvers

The adversary can attempt to recover the secret key from the public key, which is known as a key recovery attack. For FuLeeca, this is equivalent to finding any of the the rows of the secret generator matrix, which are of weight w_{key}. Alternatively, the attacker can try to forge a signature directly, without knowledge of the secret key. Forging a signature of FuLeeca is therefore equivalent to finding a codeword that satisfies both the number of required matches and the weight restriction. Hence, both attacks require solving instances of the low Lee weight codeword finding problem, which is formally defined as follows.

Problem 7 (Finding Low Lee-Weight Codewords). Given $\boldsymbol{H} \in \mathbb{F}_p^{(n-k)\times n}$ and $w \in \mathbb{N}$ find a $\boldsymbol{c} \in \mathbb{F}_p^n$ such that $\boldsymbol{c}\boldsymbol{H}^\top = \boldsymbol{0}$ and $\mathrm{wt}_L(\boldsymbol{c}) = w$.

This problem has first been studied in [27]. As usual, Problem 7, i.e. finding low-weight codewords is equivalent to the decoding problem. The decisional version of this problem has been proven to be NP-complete in [40]. Several algorithms have been proposed to solve this problem, they all belong to the family of Information Set Decoding (ISD) algorithms.

Remark 8. Note that ISD algorithms try to solve the syndrome decoding problem, that is: given a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_p^{(n-k)\times n}$, a syndrome $\boldsymbol{s} \in \mathbb{F}_p^{n-k}$ and a target weight t, they find an error vector $\boldsymbol{e} \in \mathbb{F}_p^n$, such that $\boldsymbol{H}\boldsymbol{e}^\top = \boldsymbol{s}^\top$ and $\mathrm{wt}(\boldsymbol{e}) = t$. Thus, by setting $\boldsymbol{s} = \boldsymbol{0}$, we can use such solvers to find codewords of weight t. However, note that Prange's algorithm [35] searches for a transformed syndrome $\boldsymbol{s}' = \boldsymbol{s}\boldsymbol{U}$, for some invertible $\boldsymbol{U}$ and wants the transformed syndrome to have weight t. As this is never satisfied for $\boldsymbol{s} = \boldsymbol{0}$, Prange cannot be used to find low weight codewords. However, all improvements upon Prange, such as Stern/Dumer [39, 19], MMT [31], BJMM [7] try to first enumerate the error vector in the information set and then check whether the remaining vector has the remaining weight. This can also be applied to $\boldsymbol{s} = \boldsymbol{0}$.

ISD algorithms make use of an information set of the code, where one assumes a small weight and thus constructs lists of these partial solutions. Let us quickly recall the main steps of an ISD algorithm. Given $\boldsymbol{H} \in \mathbb{F}_p^{(n-k)\times n}$, choose an information set I and bring $\boldsymbol{H}$ into a partial systematic form. For this, let J be a set of size $k+\ell$, which contains the information set I and transform $\boldsymbol{H}$ as

$$\boldsymbol{U}\boldsymbol{H}\boldsymbol{P} = \tilde{\boldsymbol{H}} = \begin{pmatrix} \mathbf{I}_{n-k-\ell} & \boldsymbol{H}_1 \\ 0 & \boldsymbol{H}_2 \end{pmatrix},$$

where $\boldsymbol{U} \in \mathbb{F}_p^{(n-k)\times(n-k)}$ is some invertible matrix and $\boldsymbol{P} \in \mathbb{F}_p^{n\times n}$ is a permutation matrix. Thus, we also split the unknown solution $\boldsymbol{c}$ into the indices J and J^c, i.e., $\boldsymbol{c}\boldsymbol{P}^\top = (\boldsymbol{c}_1, \boldsymbol{c}_2)$. Assuming that $\boldsymbol{c}_2$ has Lee weight v, we get the following two equations:

$$\boldsymbol{c}_1 + \boldsymbol{c}_2\boldsymbol{H}_1^\top = 0$$
$$\boldsymbol{c}_2\boldsymbol{H}_2^\top = 0.$$

Thus, we can first solve the second equation, $\boldsymbol{c}_2\boldsymbol{H}_2^\top = 0$ with $\mathrm{wt}_L(\boldsymbol{c}_2) = v$ as we then can easily check if the missing part $\boldsymbol{c}_1$ has the remaining Lee weight, by $\mathrm{wt}_L(\boldsymbol{c}_2\boldsymbol{H}_1^\top) = w - v$.

In [40], several algorithms have been presented to solve the smaller instance, namely using Wagner's approach of a set partitioning and using representation technique. In [13], the authors presented the amortized Wagner's approach. Finally, in [6] the authors presented an adaption of these algorithms, taking into account that a random low Lee weight codeword has the exponential weight distribution observed in [5]. In these papers, it has been observed, that the amortized BJMM approach attains the lowest computational cost, and thus we consider this algorithm to compute the security level of the proposed parameters. For the details of the algorithm, we refer to [6]. Mathematica programs to compute these computational costs are publicly available[2]. We adapted the program which computes the classical asymptotic cost c in the form $2^{c \cdot n}$, by considering the cost $c/2$ on a capable quantum computer (see [10, 13]). Since we sample the secret basis for the generator matrix using the typical Lee sets, i.e., any $x \in \mathbb{F}_p$ occurs in the sought-after error vector $\boldsymbol{e}$ in $p_{w_{key}}(x) \cdot n$ number of times, it makes sense to use this information in an ISD algorithm. However, as shown in [6], the amortized BJMM algorithm outperforms even the attempts to use restricted balls. Thus, we build our security analysis on this fastest known algorithm.

Finally, it has to be taken into account that the quasi-cyclic structure of the private key gives a polynomial speedup according to [37] and starting from a relatively large weight w_{sig} we assume that one cannot retrieve a secret basis vector. In Euclidean lattices such a technique is known as sieving and has been extensively studied [29]. Notably the works [24, 30] show that finding a codeword of low Lee weight in a quasi-cyclic code is significantly easier in case the code dimension $n/2$ is a composite number. In fact the security reduces to the codeword finding problem in a quasi-cyclic code with dimension equal to the smallest factor of $n/2$. Therefore, for all considered parameter sets in this work we choose $n/2$ to be prime.

4.2 Security against known attacks

In the multi-use scenario, an attack requires to solve the low weight codeword finding problem in the Lee metric. However, in this case, an attacker has access to several published signatures. That a previously published signature can be directly used to sign another message is prevented by setting a sufficiently high threshold for the number of required sign matches. Therefore, we consider attacks exploiting leakage via published hash/signature pairs and provide reasoning why our proposed scheme does not suffer from those attacks.

This is of particular interest since previous schemes [2, 33] allowed for attacks [36, 17], which utilize the information that is leaked by published signatures to mount attacks. In the Hamming metric a basis vector as well as the signatures have low weight, i.e., a small support. The attacks exploit the observation

[2] https://git.math.uzh.ch/isd/lee-isd/lee-isd-algorithm-complexities/-/blob/master/Lee-ISD-restricted.nb

that the supports of the published signatures, which are linear combinations of the basis elements, are correlated to the supports of the basis elements.

Such support-based attacks can not be applied to FuLeeca: in the Lee metric, basis elements of low Lee weight do not necessarily have a small support. This enables us to have secret bases which are of low Lee weight and have a relative Hamming weight of approximately 1/4. Learning this support by intersecting the supports of multiple signatures is infeasible since all published signatures have almost full Hamming weight.

To avoid leakage via published Hash/Signature pairs we integrated a specific procedure into the Signing algorithm, which we refer to as the "Concentrating" procedure. In the following we first examine the Signing algorithm without applying the specified "Concentrating" procedure. We randomly draw $k = 500$ nonces and messages and observe the corresponding outputs of the hash-function $h_1, \ldots, h_k$,

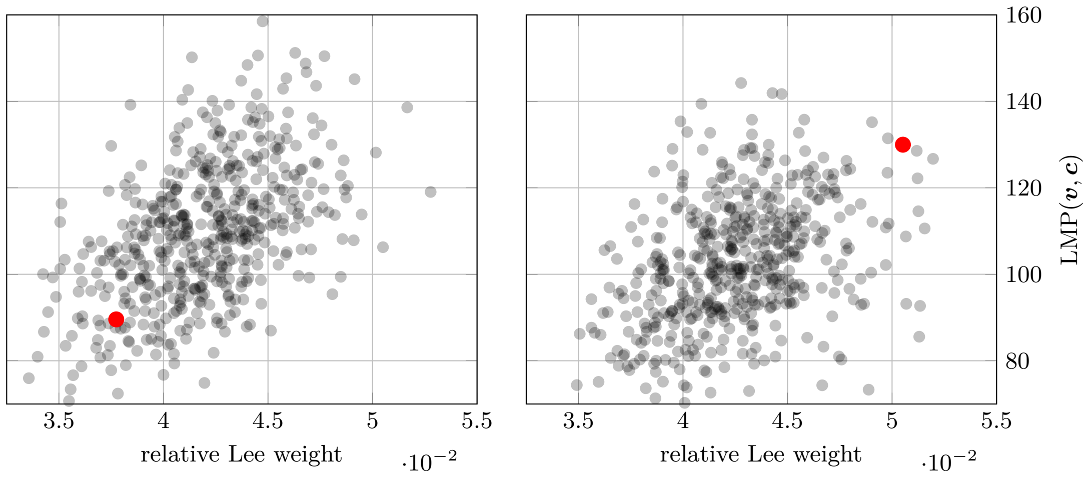

Fig. 1: Evaluation of 500 signatures for simulated hashes (i.i.d uniform) using *two different keys* (left and right) after application of "Simple Signing".

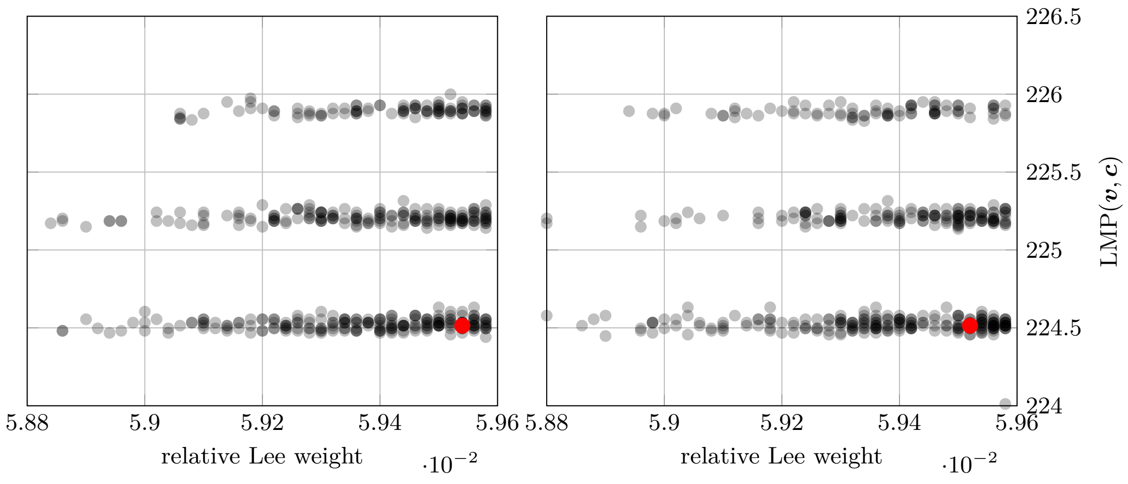

Fig. 2: Evaluation of 500 signatures for simulated hashes (i.i.d uniform) for *two different keys* after application of both "Simple Signing" and "Concentrating".

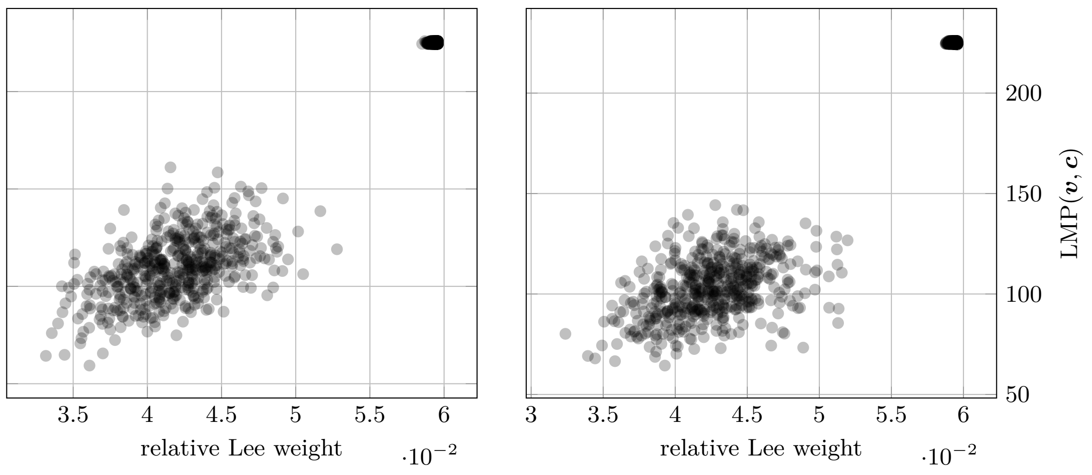

Fig. 3: Evaluation of 500 signatures for simulated hashes (i.i.d uniform) using *two different keys* both after application of the "Simple Signing" part of Algorithm 2 and as well as applying the "Concentrating" procedure (dense clusters in the upper right).

i.e., $h_\ell = \mathrm{Hash}(\mathrm{nonce}||m_\ell)$. For two different private keys we compare the Lee weights and sign matches of the corresponding signatures after just applying "Simple Signing".

Figure 1 shows the relation between the relative Lee weights and the LMP between the codeword and the target vector, which is the hash of the message. Since the signature algorithm effectively correlates the secret key and the hashes it appears to be possible to learn at least some information about the secret key based on the location of the resulting code words in this Lee weight / LMP space. The distribution of signatures for both private keys of Figure 1 show that the LMP between hash and codeword as well as the resulting Lee weights vary significantly and depend on the secret key. Since we are using two different private keys, we obtain two different signatures for each of the hashes. To exemplify this, we marked the resulting signatures before the "Concentrating" procedure for the same hash (the red dots) but using different private keys in Figure 1. Even though we do not provide a specific attack exploiting this behavior, the results show that information about the private key is leaked and can potentially be exploited to recover the secret key.

Figure 2 shows the codewords for the signatures for all hashes considered in Figure 1 after the "Concentrating" part of Algorithm 2 has been completed. The difference between the distributions for the different secret keys shall be as small as possible to minimize leakage about the secret key. Like in Figure 1 we again marked the signatures for the same hashes and different secret keys, this time after the "Concentrating" procedure in Figure 2. The results show that the "Concentrating" procedure significantly reduces the leakage observable via the relative Lee weight / LMP map.

Figure 3 provides the information observable from Figure 1 and Figure 2 within a single plot to further illustrate the effect of the "Concentrating" procedure.

Similarly, we also observe that the shape of the distribution of signatures in the Lee weight / LMP space does not appear to meaningfully depend on the distribution of the same signatures after "Simple Signing". This is demonstrated in Figure 4 and 5 where for a single key we apply "Simple Signing" to

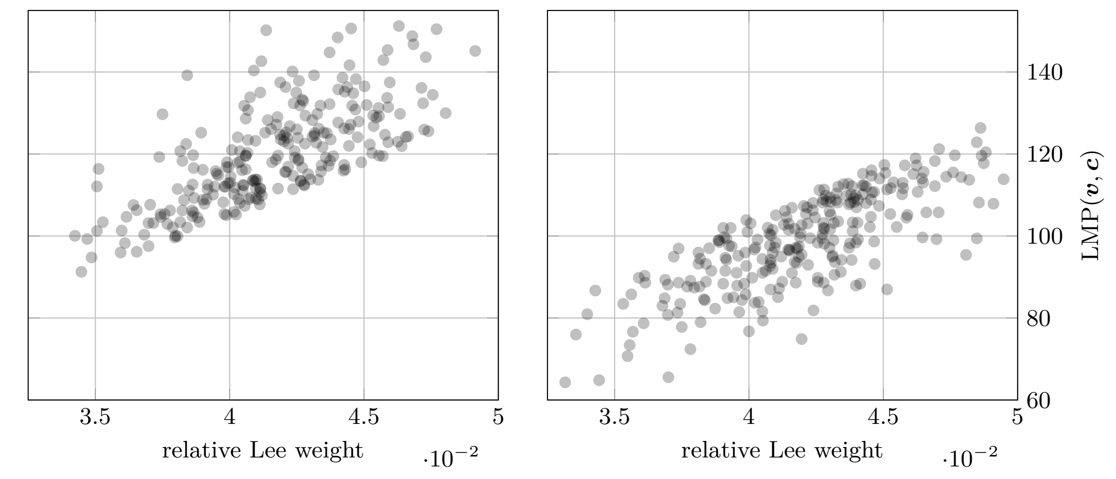

Fig. 4: Evaluation of 500 signatures for simulated hashes (i.i.d uniform) before applying the "Concentrating" procedure. Unlike the previous figures, all of the displayed signatures were created using a *single key*. The vectors are divided into two (nearly equally large) groups, where the ratio between the log probability (LMP) and the Lee weight is above average (left), respectively below average (right).

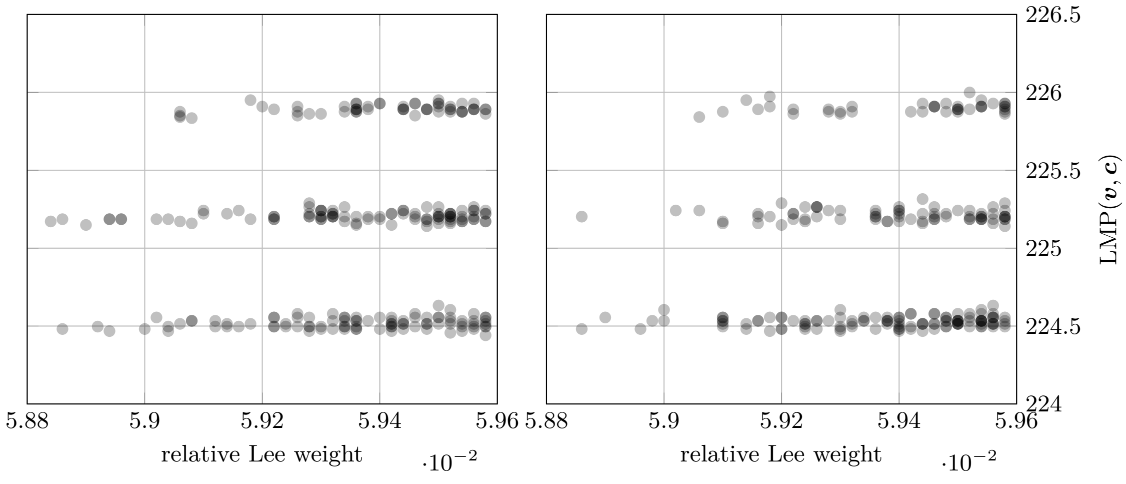

Fig. 5: The same two sets of hashes for *the same key* (as in figure 4) after applying the "Concentrating" algorithm.

the same set of hashes as before but split the signatures into two groups of almost equal size. For group one obtaining a codeword with the required LMP after application of the "Concentrating" procedure is expected to be easier than for group two since in terms of the ratio between the log probability and the Lee weight all of these are above average, while group two is below average. In fact, the percentage of hashes in group two that lead to a valid signature in the end is slightly lower than for group one, however this observation is to be expected for effectively every private key and thus this does not reveal any useful information about any chosen key in particular.

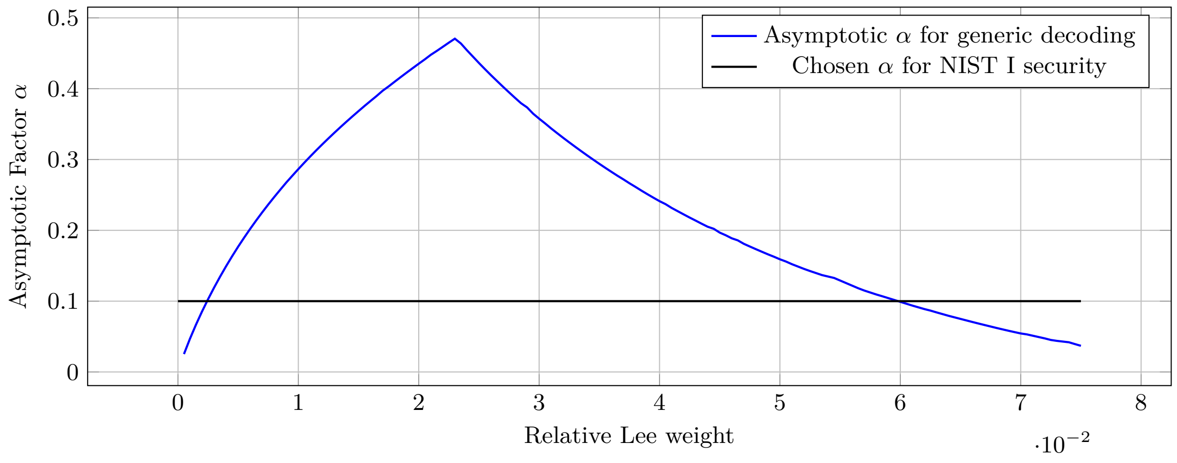

Fig. 6: Asymptotic Quantum Cost of Amortized Lee BJMM for $p = 251$

4.3 Security of the chosen parameters

We assume that the used Hash functions are cryptographically secure.

The best known attack to find a codeword of low Lee weight given our public key $\boldsymbol{G}_{pub}$ is Information Set Decoding using the quantum amortized BJMM algorithm in the Lee metric. Therefore, the complexity of this attack is used to determine the parameters. The Lee weight w_{key} of the secret key, i.e., $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_p^{n/2}$ with $\text{wt}_L(\boldsymbol{a}) = \text{wt}_L(\boldsymbol{b}) = w_{\text{key}}$ and the Lee weight w_{sig} of the codeword connected to the signature, i.e., $\boldsymbol{v} \in \mathbb{F}_p^n$ with $\text{wt}_L(\boldsymbol{v}) = w_{\text{sig}}$ are chosen as follows: since we want both, key recovery attacks and forgery attacks, to have a cost of at least $2^{\lambda'} = 2^{n \cdot \alpha}$, we plot the asymptotic cost α of the ISD algorithm for relative Lee weights in the interval $[0, 1]$, as shown in Figure 6. For the chosen $p = 251$, the minimum Lee distance on the GV bound is at $\delta = 0.023$. This is exactly the peak of the graph, with an asymptotic cost factor of 0.47 in the binary complexity exponent. This allows us to choose the two intersection points of the cost curve with $y = \frac{\lambda'}{n}$ as the key and signature weights. That is w_{key} is chosen as the first intersection point and w_{sig} is chosen as the second intersection point. For example, the parameter choice $p = 251, n = 778, w_{sig} = 0.0588, w_{key} = 0.00255$ leads to the desired quantum cost of 2^{80}, since $\alpha(w_{sig}) = \alpha(w_{key}) = 0.1042$. This simple routine is performed for all proposed parameter sets.

5 Efficiency & Performance

Due to the quasi-cyclic structure of the private matrix $\boldsymbol{G}_{sec}$ it is sufficient to store only one of its rows. Therefore, the size of the private key is in the order $\mathcal{O}_p(n)$, where the constant depends on the parameter p.

We take a conservative choice for the NIST security levels [28], as shown in Table 1. The chosen parameters and associated data sizes for the NIST categories I, III and V are given in Table 2. The signature size is the size averaged over 10k generated compressed signatures. For compression, we have adapted the mechanisms as used in the Falcon signature scheme. These compressed signatures can be padded to have a fixed size. Notice that the signature sizes already include the size of the nonces. As

Table 1: Conservative NIST Categories

NIST Security Level	Classical Cost	Quantum Cost
I	160	80
III	224	112
V	288	144

Table 2: Parameters for the proposed signature scheme FuLeeca. All sizes are given in Bytes.

q	n	ω_{sig}	ω_{key}	NIST category	secret key size	public key size	sign. size
251	778	0.059	0.0025	I	778	389	276
251	1154	0.060	0.0023	III	1154	577	408
251	1538	0.061	0.0021	V	1538	769	540

proposed in [20], it is possible to compress the signatures resulting from Algorithm 2 to achieve the signature sizes given in Table 2. Note that the chosen parameters are not optimized yet and thus, we expect further improvements.

A comparison with SPHINCS+ [1], Dilithium [18] and Falcon [23] shows that our scheme provides parameters that outperform all three NIST selected schemes in terms of total bandwidth (public key + signature size). Furthermore, we outperform all known code-based signature schemes in terms of public key and signature size.

6 Conclusion

In this paper, we proposed a Hash-and-Sign signature scheme based on the problem of finding low Lee weight codewords. Taking known statistical attacks into account, we refined the simple signing process to render the scheme multiple-use. The scheme can be efficiently implemented as it only uses simple arithmetics and is able to achieve extremely short signatures of 276 bytes and public keys of 389 bytes

Table 3: Comparison to other post-quantum signature schemes for the NIST category I security level (and NIST level II for Dilithium). All sizes are given in Bytes.

Scheme	public key size	sign. size	Total data size
FuLeeca	389	276	665
WAVE	>2 000 000	2100	>2 002 100
Falcon	897	666	1563
SPHINCS+	32	7856	7888
Dilithium	1312	2420	3732

for the NIST category I security level. This compares favorably to state-of-the-art lattice-based and code-based schemes.

Bibliography

[1] Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. SPHINCS$^+$, submission to the NIST post-quantum project, v.3. 2020.

[2] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. Using lDGM codes and sparse syndromes to achieve digital signatures. In *Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings 5*, pages 1–15. Springer, 2013.

[3] Alexander Barg. Complexity Issues in Coding Theory. Technical Report TR97-046, Electronic Colloquium on Computational Complexity (ECCC), October 1997. ISSN: 1433-8092.

[4] Barg, Alexander. Some New NP-Complete Coding Problems. *Problemy Peredachi Informatsii*, 30(3):23–28, 1994.

[5] Jessica Bariffi, Hannes Bartz, Gianluigi Liva, and Joachim Rosenthal. On the properties of error patterns in the constant lee weight channel. In *International Zurich Seminar on Information and Communication (IZS 2022). Proceedings*, pages 44–48. ETH Zurich, 2022.

[6] Jessica Bariffi, Karan Khathuria, and Violetta Weger. Information set decoding for lee-metric codes using restricted balls. In *Code-based Cryptography 10th International Workshop, CBCrypto 2022.* Springer.

[7] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1+1=0$ improves information set decoding. In *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pages 520–536. Springer, 2012.

[8] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with RSA and Rabin. In *Eurocrypt*, volume 96, pages 399–416. Springer, 1996.

[9] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, May 1978.

[10] Daniel J Bernstein. Grover vs. McEliece. In *Post-Quantum Cryptography: Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings 3*, pages 73–80. Springer, 2010.

[11] Mainak Bhattacharyya and Ankur Raina. A quantum algorithm for syndrome decoding of classical error-correcting linear block codes. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pages 456–461, December 2022.

[12] Eimear Byrne, Anna-Lena Horlemann, Karan Khathuria, and Violetta Weger. Density of free modules over finite chain rings. *Linear Algebra and its Applications*, 651:1–25, 2022.

[13] André Chailloux, Thomas Debris-Alazard, and Simona Etinski. Classical and Quantum algorithms for generic Syndrome Decoding problems and applications to the Lee metric, 2021. Report Number: 552.

[14] Jinkyu Cho, Jong-Seon No, Yongwoo Lee, Zahyun Koo, and Young-Sik Kim. Enhanced pqsigRM: Code-based digital signature scheme with short signature and fast verification for post-quantum cryptography. *Cryptology ePrint Archive*, 2022.

[15] Nicolas T Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in CryptologyASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 157–174. Springer, 2001.

[16] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I*, pages 21–51. Springer, 2019.

[17] Jean-Christophe Deneuville and Philippe Gaborit. Cryptanalysis of a code-based one-time signature. *Designs, Codes and Cryptography*, 88:1857–1866, 2020.

[18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium algorithm specifications and supporting documentation (version 3.1). 2021.

[19] Il'ya Isaakovich Dumer. Two decoding algorithms for linear codes. *Problemy Peredachi Informatsii*, 25(1):24–32, 1989.

[20] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Shorter hash-and-sign lattice-based signatures. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*, pages 245–275. Springer, 2022.

[21] Jean-Charles Faugere, Valérie Gauthier-Umana, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high-rate McEliece cryptosystems. *IEEE Transactions on Information Theory*, 59(10):6830–6844, 2013.

[22] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto*, volume 86, pages 186–194. Springer, 1986.

[23] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON: fast-fourier lattice-basd compact signatures over NTRU, specification v1.2. 2020.

[24] Craig Gentry. Key recovery and message attacks on ntru-composite. In *Advances in CryptologyEUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20*, pages 182–194. Springer, 2001.

[25] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[26] Danilo Gligoroski, Simona Samardjiska, Håkon Jacobsen, and Sergey Bezzateev. Mceliece in the world of escher. *Cryptology ePrint Archive*, 2014.

[27] Anna-Lena Horlemann-Trautmann and Violetta Weger. Information set decoding in the lee metric with applications to cryptography. *Advances in Mathematics of Communications*, 15(4), 2021.

[28] Kyungbae Jang, Anubhab Baksi, Hyunji Kim, Gyeongju Song, Hwajeong Seo, and Anupam Chattopadhyay. Quantum analysis of aes. *Cryptology ePrint Archive*, 2022.

[29] Elena Kirshanova and Thijs Laarhoven. Lower bounds on lattice sieving and information set decoding. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*, pages 791–820. Springer, 2021.

[30] Carl Löndahl, Thomas Johansson, Masoumeh Koochak Shooshtari, Mahmoud Ahmadian-Attari, and Mohammad Reza Aref. Squaring attacks on mceliece public-key cryptosystems using quasi-cyclic codes of even dimension. *Designs, Codes and Cryptography*, 80:359–377, 2016.

[31] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\mathcal{O}(2^{0.054n}$. In *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*, pages 107–124. Springer, 2011.

[32] Dustin Moody and Ray Perlner. Vulnerabilities of McEliece in the World of Escher. In *Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings 7*, pages 104–117. Springer, 2016.

[33] Edoardo Persichetti. Efficient one-time signatures from quasi-cyclic codes: A full treatment. *Cryptography*, 2(4):30, 2018.

[34] Aurélie Phesso and Jean-Pierre Tillich. An efficient attack on a code-based signature scheme. In *Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings 7*, pages 86–103. Springer, 2016.

[35] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[36] Paolo Santini, Marco Baldi, and Franco Chiaraluce. Cryptanalysis of a one-time code-based digital signature scheme. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2594–2598. IEEE, 2019.

[37] Nicolas Sendrier. Decoding One Out of Many. In *Post-Quantum Cryptography*, Lecture Notes in Computer Science, pages 51–67, Berlin, Heidelberg, 2011. Springer.

[38] P.W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, November 1994.

[39] Jacques Stern. A method for finding codewords of small weight. *Coding theory and applications*, 388:106–113, 1989.

[40] Violetta Weger, Karan Khathuria, Anna-Lena Horlemann, Massimo Battaglioni, Paolo Santini, and Edoardo Persichetti. On the hardness of the Lee syndrome decoding problem. *Advances in Mathematics of Communications*, pages 0–0, April 2022. Publisher: Advances in Mathematics of Communications.