Cryptanalysis of an Efficient Signature Based on Isotropic Quadratic Forms

Henry Bambury[1,2] and Phong Q. Nguyen[1]

[1] DIENS, École normale supérieure, PSL University, CNRS, Inria, Paris, France
[2] DGA, Paris, France

Abstract. We present a key-recovery attack on DEFI, an efficient signature scheme proposed recently by Feussner and Semaev, and based on isotropic quadratic forms, borrowing from both multivariate and lattice cryptography. Our lattice-based attack is partially heuristic, but works on all proposed parameters: experimentally, it recovers the secret key in a few minutes, using less than ten (message,signature) pairs.

Keywords: Cryptanalysis · Digital Signatures · Lattices · Multivariate Cryptography · Quadratic Forms

1 Introduction

The lattice-based signature schemes Dilithium [8] and Falcon [15] have been selected by the NIST [22] as the first standards for post-quantum cryptography. But this post-quantum security comes at a cost: the size of both the public key and the signature of Dilithium and Falcon are significantly bigger than for ECDSA and RSA. It would be useful to have more efficient post-quantum signature schemes and/or based on different assumptions: this motivated the NIST to open a call for additional digital signature proposals [21] in 2022. In that call, Feussner and Semaev submitted the lattice-based signature scheme EHTv3v4 [12], which currently remains unbroken after a fix. Very recently [13], the same authors proposed a very different and much more efficient scheme, called DEFI, on the NIST pqc mailing list: with a 800-byte public key and a 432-byte signature, DEFI is more efficient than both Dilithium and Falcon, and beats all additional NIST submissions except for SQISign in (public key + signature) size [23]. Even with a non-optimised implementation, DEFI's signature and verification times seem to compare favourably to all proposed signatures [5]. DEFI is a peculiar scheme borrowing from both multivariate cryptography and lattice-based cryptography: its security is based on the hardness of solving systems of quadratic equations over the integers and a polynomial ring R such as $\mathbb{Z}[X]/(X^{64}+1)$. In its general form, this problem is known to be NP-hard, and therefore the authors of DEFI assumed it hard in the worst case, but DEFI uses special instances of the problem, which might be much easier to solve.

More precisely, a DEFI private key is a solution to a small system of quadratic equations over R, determined by the DEFI public key. Because R is a polynomial

ring, this small system can be transformed into a large system of quadratic equations over $\mathbb{Z}$, which in general would be an NP-hard problem. DEFI is a hash-and-sign probabilistic scheme: the signature of a hashed message h is simply a randomly-generated solution to a small system of quadratic equations over R, in such a way that the first entry of the solution vector is h, and the other entries depend on the choice of a nonce, a one-time key required for each signature generation. Surprisingly, DEFI does not use modular arithmetic nor finite fields: all operations are in the polynomial ring R, and this was a security argument in [13]. Feussner and Semaev analysed [13] several attacks on DEFI to argue that their scheme DEFI was immune against Gröbner basis attacks and lattice attacks. They proposed a 64-bit numerical challenge, and concrete parameters for which they conjectured a 128-bit security level.

Our results. We show that DEFI is completely insecure: experimentally, less than ten (message,signature) pairs are sufficient to recover the secret key in a few minutes, for all parameters proposed in [13], including the 64-bit challenge.

Disclaimer. In reaction to the break of DEFI presented here, the authors of the scheme proposed DEFIv2[14], a new and improved version of the signature scheme, boasting similarly impressive performances. The present article was written before DEFIv2 was announced and thus only focuses on the cryptanalysis of DEFI. The attack presented here does not seem to be directly applicable to DEFIv2.

Technical overview. The starting point of our attack is that each DEFI signature leaks information on the secret key, and we exploit that leakage: each signature provides a linear equation over R, whose solutions are related to the secret key. By collecting enough signatures, we obtain a linear system of equations over R: this gives rise to a lattice whose rank is independent of the number of signatures used, but for which we know an unusually short vector related to the private key and the secret nonces which were used to generate each signature. The more signatures we use, the more unusually short the secret vector becomes, without affecting the rank of the lattice.

By reducing this lattice, we heuristically obtain this unusually short vector. At this point, we cannot yet recover the secret key. However, it allows us to derive a new linear system of equations over R: this gives rise to a second lattice, whose rank depends on the number of signatures. We know that this lattice contains another very short vector, which is directly related to the private and the secret nonces which were used to generate each signature. Again, if we take more signatures into account, then this second very short secret vector becomes even shorter, relatively to what one would expect from a typical lattice. However, the lattice rank increases with the number of signatures used in this second stage, making lattice reduction increasingly expensive. We circumvent this issue by noting the existence of and heuristically recovering an unusually dense sublattice

of much smaller rank that contains the targeted second secret short vector. We then reduce the recovered sublattice to obtain the second secret.

Together, the two unusually short vectors that were recovered provide a final system of linear equations over $\mathcal{R}$, whose solutions are exactly the secret key and its rotations. If enough signatures are given, we obtain a linear system over $\mathbb{Z}$ for which there are many more equations than unknowns: this recovers the secret key in polynomial time by linear algebra, provided that the linear system is full-rank. Alternatively, the structure of the two unusually short vectors allows us to recover the secret key efficiently by an ad-hoc process, based on the equation relating the public key and the secret key.

To summarise, there are three stages in the attack: the first two stages use lattice reduction to find extremely short vectors, but the rank of the lattice used in the first stage is independent of the number of signatures used. The final third stage recovers the private key without lattice techniques.

Related work. The authors of DEFI [13] also considered lattice attacks, but showed that their attacks failed. However, their attacks were different from our attack, even though their attacks exploited the same equations that we are using. These attacks failed because of two reasons. The first reason is that [13] used a different lattice, whose dependence on the secret key was much less useful: in this lattice, there was apparently no unusually short lattice vector related to the secret key. The second reason is that [13] only considered attacks using a single signature.

Roadmap. Section 2 introduces notations and recalls useful facts. Section 3 presents the DEFI signature scheme. We describe our attack in Section 4, give some elements of justification in Section 5, and present our experimental results in Section 6.

2 Preliminaries

General notations. Vectors are written in bold lowercase $\mathbf{v}$. The Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^n$ is denoted $\|\mathbf{v}\|$. Throughout this paper, we use row representation of matrices. For a set R and a positive integer $n \in \mathbb{Z}_{>0}$, $M_n(R)$ denotes the set of $n \times n$ matrices with entries in R, and $\mathrm{diag}(\alpha_1, \ldots, \alpha_n)$ denotes the diagonal matrix of $M_n(R)$ with coefficients α_i. We use $[n]$ as a notation for $\{1, \ldots, n\}$. Matrices and vectors will be written in bold font. If $z_1 \in R$ and $\mathbf{z} = (z_2, \ldots, z_n) \in R^{n-1}$, we will use $(z_1||\mathbf{z})$ to denote $(z_1, \ldots, z_n) \in R^n$.

Coefficient embedding. Let $R = \mathbb{Z}[X]/(q)$ be a polynomial ring, where q is a degree m monic irreducible polynomial. Elements of R are represented by their *coefficient embedding*:

$$\mathrm{coef} : \begin{cases} R & \to \mathbb{Z}^m \\ a = \sum_{i=0}^{m-1} a_i X^i & \mapsto (a_0, \ldots, a_{m-1}) \end{cases}.$$

When speaking of the shortness of $a \in R$, we mean the shortness of the corresponding vector $\text{coef}(a) \in \mathbb{Z}^m$. This allows us to naturally extend the L_2 and L_∞ norms $\|\cdot\|$ and $\|\cdot\|_\infty$ to R. More generally, we extend both norms to direct products of R by considering concatenations of the coefficient vectors.

Short elements in polynomial rings. There are many ways to sample short elements in R. While some schemes might sample each coefficients independently and with the same distribution, others like NTRU or DEFI fix a number of non-zero coordinates λ, which they choose uniformly at random, and uniformly sample all non-zero coordinates in a small set of values, such as $\{\pm 1, \pm 2\}$ in DEFI, we note D_u the distribution from which u is sampled.

Lattices. A *lattice* L is a discrete subgroup of $\mathbb{R}^m$. Alternatively, we can define a lattice as the set $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^n x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ of all integer combinations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$. This sequence of vectors is known as a *basis* of the lattice L. All the bases of L have the same number n of elements, called the dimension or rank of L, and the n-dimensional volume of the parallelepiped $\{\sum_{i=1}^n a_i \mathbf{b}_i \ : \ a_i \in [0,1)\}$ they generate. We call this volume the covolume, or determinant, of L, and denote it by $\text{vol}(L)$. The lattice L is said to be *full-rank* if $n = m$. We denote by $\lambda_1(L)$ the first minimum of L, defined as the norm of a shortest nonzero vector of L.

Orthogonalisation. For a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, and an index $1 \leq i \leq n$, we denote by π_i the orthogonal projection on $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp$. The *Gram-Schmidt orthogonalisation* (GSO) of the basis B is defined as the orthogonal sequence of vectors $B^\star = (\mathbf{b}_1^\star, \ldots, \mathbf{b}_n^\star)$, where $\mathbf{b}_i^\star := \pi_i(\mathbf{b}_i)$. When we speak of the (log) Gram-Schmidt profile, we refer to the plot of the quantities $(\log \|\mathbf{b}_1^\star\|, \ldots, \log \|\mathbf{b}_n^\star\|)$. It will represent how well reduced the lattice basis is.

Random lattices and Gaussian heuristic. The space $X_n = \text{SL}_n(\mathbb{R})/\text{SL}_n(\mathbb{Z})$ of covolume 1 real lattices has a unique $\text{SL}_n(\mathbb{Z})$-invariant Haar probability measure, that defines the mathematically correct way of thinking about a random lattice. The expected value for such a random lattice's first minimum is sometimes referred to as the Gaussian heuristic radius for lattices. For a rank n lattice L, we denote $\text{GH}(L) := \text{vol}(B_n)^{-1/n}\text{vol}(L)^{1/n} = (1 + o_n(1))\sqrt{\frac{n}{2\pi e}}\text{vol}(L)^{1/n}$, where B_n is the n-dimensional L2 ball of radius 1.

Lattice reduction. The problem of recovering a shortest vector in a lattice is called the *Shortest Vector Problem* (SVP). Various algorithms exist that solve approximated versions of SVP. To name a couple, we have the LLL algorithm [18,24] that runs in polynomial time for exponentially large approximation factors, as well as the BKZ hierarchy of algorithms [25,4] that presents a trade-off between time and approximation factor.

3 The DEFI Signature Scheme

In [13], Feussner and Semaev propose a new digital signature scheme called DEFI, based on solving systems of quadratic Diophantine equations over the rational integers.

3.1 Formal Definition of the Scheme

Let $q \in \mathbb{Z}[X]$ be a monic irreducible polynomial, and $R = \mathbb{Z}[X]/(q)$ its associated polynomial ring, where (q) denotes the ideal of $\mathbb{Z}[X]$ generated by q. Let $\mathbf{J} = \mathrm{diag}(1, 1, -1, -1) \in M_4(R)$. For $\mathbf{A} \in M_n(R)$, we define $f_{\mathbf{A}}$ by $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ for $\mathbf{x} \in R^n$. In [13], the authors seem to consider a wider array of matrices $\mathbf{J} = \mathrm{diag}(\pm 1, \ldots, \pm 1) \in M_n(R)$, where n can vary, but the instantiation of their scheme heavily relies on the specific choices $n = 4$ and $\mathbf{J} = \mathrm{diag}(1, 1, -1, -1)$. Unless design choices are made radically different for another choice of $\mathbf{J}$, an adapted version of our attack would still apply.

Private Key. The *private key* is a matrix $\mathbf{B} \in M_4(R)$, defined blockwise as

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_1 & \mathbf{0} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix},$$

where $\mathbf{B}_{21} \in R^3$ and $\mathbf{B}_{22} \in M_3(R)$. $\mathbf{B}$ should be invertible and $\mathbf{B}_{21}, \mathbf{B}_{22}$ and $\mathbf{B}_{22}^{-1}$ are taken with small norm (*i.e.* elements are polynomials of R with small coefficients). Note that invertibility of $\mathbf{B}$ implies it is unimodular. We refer to [13] for the precise generation procedure for $\mathbf{B}$. While the condition on the size of $\mathbf{B}$ in [13] is slightly different, we will assume that $\|\mathbf{B}_{21}\|_\infty < \delta_{\mathbf{B}_{21}}$ and $\|\mathbf{B}_{22}\|_\infty < \delta_{\mathbf{B}_{22}}$, where $\delta_{\mathbf{B}_{21}}$ and $\delta_{\mathbf{B}_{22}}$ are all parameters chosen in Table 1. We would like to stress that this is not an important change, as it does not make the scheme less secure and only helps increase the readability of our analysis. We refer to Figure 1 for experimental confirmation that all the private keys that were generated from the reference implementation [11] satisfy our bounds.

Public Key. The *public key* is the matrix

$$\mathbf{C} = \mathbf{B}^T\mathbf{J}\mathbf{B}.$$

Again, the authors of DEFI choose to reject matrices $\mathbf{C}$ that have large entries, as this allows for shorter public keys. We will not use this fact in our attack.

Signature generation. The following procedure is used to sign a message μ from a private key $\mathbf{B}$. The full pseudo-code is described in Algorithm 2.

1. A message μ is first hashed into $h := H(\mu) \in R$.
2. A special trapdoor procedure constructs a $\mathbf{z} = \big(h \,||\, \mathbf{z}'\big)$ such that $f_{\mathbf{J}}(\mathbf{z}) = 0$. This step is described in Algorithm 1. It completes the hashed message with a random nonce in such a way that the resulting vector is isotropic with respect to $\mathbf{J}$.
3. The signature is $\mathbf{y} := \mathbf{B}_{22}^{-1}(\mathbf{z}' - \mathbf{B}_{21}h)$.

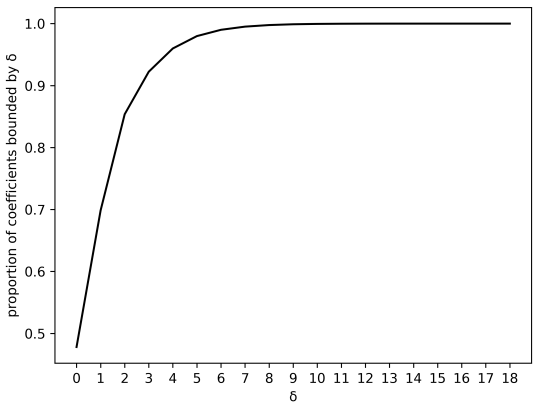


Fig. 1: Proportion of *coefficient embedding* coefficients of $\mathbf{B}$ that have absolute value less than an integer δ, out of 1000 DEFI-128 samples. We note that all coefficients were smaller than $\delta_{\mathbf{B}_{22}} = 18$.

Algorithm 1 GenerateZ($\cdot$)

Input: $z_1 \in R$.
Output: $\mathbf{z}' \in R^3$ such that $f_{\mathbf{J}}((z_1 \mid\mid \mathbf{z}')) = 0$.
1: $u_1, v_2 \leftarrow D_u$ $\{u_1, v_2 \in R\}$
2: $v \leftarrow v_2(1 - u_1^2)$ $\{v \in R\}$
3: $u_2 \leftarrow 2v_2$ $\{u_2 \in R\}$
4: $z_2 \leftarrow v + u_2 u_1^2 - z_1 u_1$ $\{z_2 \in R\}$
5: $z_3 \leftarrow v + z_1 u_1$ $\{z_3 \in R\}$
6: $z_4 \leftarrow u_1 u_2 - z_1$ $\{z_4 \in R\}$
7: $\mathbf{z}' \leftarrow (z_2 \mid\mid z_3 \mid\mid z_4)$ $\{\mathbf{z}' \in R^3\}$
8: Return $\mathbf{z}'$

Signature verification. Verification is described in Algorithm 3. It consists of the following two steps:

1. The message μ is hashed into $h := H(\mu)$.
2. The signature is *accepted* if and only if $f_{\mathbf{C}}((h \mid\mid \mathbf{y})) = 0$, where $\mathbf{y}$ is the signature and $\mathbf{C}$ is the public key.

3.2 Correctness of the Scheme

Let $(\mu, \mathbf{y})$ be a valid signature obtained using the secret key $\mathbf{B}$, $h = H(\mu)$ and $\mathbf{C}$ the associated public key. $\mathbf{y} = \mathbf{B}_{22}^{-1}(\mathbf{z}' - \mathbf{B}_{21}h)$ where $\mathbf{z}'$ is the output of GenerateZ(h). Let $\mathbf{z} = (h \mid\mid \mathbf{z}')$, and $\mathbf{x} = (h \mid\mid \mathbf{y})$. Then

$$f_{\mathbf{C}}(\mathbf{x}) = \mathbf{x}^T\mathbf{C}\mathbf{x} = (\mathbf{B}\mathbf{x})^T\mathbf{J}(\mathbf{B}\mathbf{x}) = f_{\mathbf{J}}(\mathbf{B}\mathbf{x}) = f_{\mathbf{J}}(\mathbf{z}).$$

Algorithm 2 DEFI signature generation

Input: A message μ and a private key $\mathbf{B} \in M_4(R)$.
Output: A valid signature $\mathbf{y} \in R^3$.
1: $h \leftarrow H(\mu)$ $\{h \in R\}$
2: $\mathbf{z}' \leftarrow \text{GenerateZ}(h)$ $\{\mathbf{z}' \in R^3\}$
3: $\mathbf{y} \leftarrow \mathbf{B}_{22}^{-1}(\mathbf{z}' - \mathbf{B}_{21}h)$ $\{\mathbf{y} \in R^3\}$
4: Return $\mathbf{y}$

Algorithm 3 DEFI signature verification

Input: A message μ, a signature $\mathbf{y} \in R^3$ and a public key $\mathbf{C} \in M_4(R)$.
Output: *Accept* if the signature is correct, *Reject* otherwise.
1: **if** $f_{\mathbf{C}}(\big(h \parallel \mathbf{y}\big)) = 0$ **then**
2: Return *Accept*
3: **else**
4: Return *Reject*
5: **end if**

Indeed,

$$\mathbf{Bx} = \begin{pmatrix} \mathbf{I}_1 & \mathbf{0} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix} \begin{pmatrix} h \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} h \\ \mathbf{B}_{21}h + \mathbf{B}_{22}\mathbf{y} \end{pmatrix} = \begin{pmatrix} h \\ \mathbf{z}' \end{pmatrix} = \mathbf{z}. \tag{1}$$

In order to prove the correctness of the scheme, we need to prove that Algorithm 1 produces a vector that is isotropic with respect to $\mathbf{J}$ once is is concatenated with the hash of the message. We use the notations of Algorithm 1 for the coordinates of $\mathbf{z}'$:

$$\begin{aligned} f_{\mathbf{J}}(\mathbf{z}) = \mathbf{z}^T\mathbf{J}\mathbf{z} &= z_1^2 + z_2^2 - z_3^2 - z_4^2 \\ &= (z_1 + z_4)(z_1 - z_4) + (z_2 + z_3)(z_2 - z_3) \\ &= (u_1u_2)(2z_1 - u_1u_2) + (2v + u_2u_1^2)(u_2u_1^2 - 2z_1u_1) \\ &= u_1u_2(2z_1 - u_1u_2) + (u_2(1 - u_1^2) + u_2u_1^2)(u_2u_1^2 - 2z_1u_1) \\ &= u_1u_2(2z_1 - u_1u_2 + u_1u_2 - 2z_1) = 0. \end{aligned}$$

3.3 Parameter Choice

DEFI comes in two flavours, a challenge version called DEFI-64, and a reference version DEFI-128 that was claimed to provide 128 bits of security. The ring R is defined according to the parameter m as $\mathbb{Z}[X]/(X^m + 1)$. The distribution D_u samples λ_u non-zero coordinates and uniformly assigns them a number from $\{\pm 1, \pm 2\}$.

	m	λ_u	$\delta_{\mathbf{B}_{21}}$	$\delta_{\mathbf{B}_{22}}$
DEFI-64	32	15	2	15
DEFI-128	64	35	2	18

Table 1: Parameters for DEFI

4 Attacking DEFI

Lattice attacks on DEFI were already discussed in [13, Section *IV.E.*]. The authors of the scheme observe that

$$z_2 + z_3 = u_2 \tag{2}$$
$$z_1 + z_4 = u_1 u_2, \tag{3}$$

where $z_2 = b_{21}h + b_{22}y_2 + b_{23}y_3 + b_{24}y_4$ and $z_3 = b_{31}h + b_{32}y_2 + b_{33}y_3 + b_{34}y_4$, using the notation $\mathbf{B} = (b_{ij})_{i,j\in[4]}$. To exploit Equation 2, they argue that recovering the desired vector $\mathbf{b} = (b_{21}, b_{22}, b_{23}, b_{24}, b_{31}, b_{32}, b_{33}, b_{34}, u_2) \in R^9$ as a SVP solution in the lattice

$$L = \{\mathbf{x} \in R^9 : x_1h + x_2y_2 + x_3y_3 + x_4y_4 + x_5h + x_6y_2 + x_7y_3 + x_8y_4 - x_9 = 0\}$$

should be difficult, as the experimental norm of $\mathbf{b}$ is much larger than the Gaussian heuristic for L, implying that *if L behaves like a typical random lattice* then its shortest vector would be much shorter than $\mathbf{b}$. Their analysis is not exhaustive and overlooks a number of things. Most notably, L only exploits the information obtained from a single signature, and focuses on recovering the coefficients of B directly, whereas other more intermediate secrets might be easier to obtain through lattice attacks. No further justification is given to assess that it is sound to use the Gaussian heuristic for the length of the shortest vector here.

In our attack, we assume that the attacker has access to k signatures and use $x^{(i)}$ to denote the value of the parameter x corresponding to the i-th signature. We would like to emphasise that the attack is heuristic (as is often in the case in lattice-based cryptanalysis). Our analysis is presented in Section 5. Experiments are later discussed in Section 6.

4.1 First Step: Recovering u_2

The first step of the attack exploits the fact that each signature contributes to some leakage in order to recover half of the randomness used to generate the isotropic vectors that are used in the signing process (recall that Algorithm 1 samples u_1 and v_1, here we recover $u_2 = 2v_2$), as well as some partial information on coefficients of the secret matrix $\mathbf{B}$. Obervation 1 below summarises exactly what we get from this first step.

Observation 1 (Informal) *If an attacker has access to a large enough number k of signatures $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(k)}$ signed with the same private key $\mathbf{B}$, then it can recover $b_{21} + b_{31}$, $b_{22} + b_{32}$, $b_{23} + b_{33}$, $b_{24} + b_{34}$ and all $u_2^{(i)}$ in time polynomial in $m = \dim(R)$ and the size of the entries.*

We aim to recover this secret information by using lattice reduction, to find a short vector in a lattice. We first define our lattice L_1, show how to construct it efficiently from public information, and finally we write down a short vector $\mathbf{s}_1 \in L_1$.

Definition 1. *Let $k \in \mathbb{Z}_{>0}$. Assuming the vectors $(h^{(i)}, y_2^{(i)}, y_3^{(i)}, y_4^{(i)}) \in R^4$ for $i \in \mathbb{Z}_{>0}$ form a sequence of signatures obtained using the same private key, we define the following lattice*

$$L_1 = \left\{(\alpha, \beta, \gamma, \delta, \varepsilon^{(1)}, \ldots, \varepsilon^{(k)}) \in R^{4+k} : \forall i, \varepsilon^{(i)} = \alpha h^{(i)} + \beta y_2^{(i)} + \gamma y_3^{(i)} + \delta y_4^{(i)}\right\}.$$

We note that L_1 can be seen interchangeably as an R-module or as a Euclidean lattice.

Proposition 1 (Properties of L_1). *Let $k \in \mathbb{Z}_{>0}$. Then the following statements are true:*

- *L_1 has rank $4m$;*
- *L_1 has ambient dimension $(k+4)m$;*
- *A basis of L_1 can be efficiently computed from the first k signatures.*

Proof. By definition, L_1 is generated by the $4m$ following vectors

$$\begin{gathered}\left(X^j, 0, 0, 0, X^j h^{(1)}, \ldots, X^j h^{(k)}\right);\\ \left(0, X^j, 0, 0, X^j y_2^{(1)}, \ldots, X^j y_2^{(k)}\right);\\ \left(0, 0, X^j, 0, X^j y_3^{(1)}, \ldots, X^j y_3^{(k)}\right);\\ \left(0, 0, 0, X^j, X^j y_4^{(1)}, \ldots, X^j y_4^{(k)}\right),\end{gathered}$$

for $j \in [m]$. This gives an efficiently computable generating set of vectors of L_1 of size $4m$. The first four R-coordinates are all linearly independent, therefore L_1 has rank exactly $4m$. □

Let

$$\mathbf{s}_1 = (b_{21} + b_{31}, b_{22} + b_{32}, b_{23} + b_{33}, b_{24} + b_{34}, u_2^{(1)}, \ldots, u_2^{(k)}).$$

Recall that for all $i \in [k]$ we have as in Equation 1,

$$\begin{pmatrix} b_{22} & b_{23} & b_{24} \\ b_{32} & b_{33} & b_{34} \end{pmatrix} \begin{pmatrix} y_2^{(i)} \\ y_3^{(i)} \\ y_4^{(i)} \end{pmatrix} = \begin{pmatrix} z_2^{(i)} \\ z_3^{(i)} \end{pmatrix} - h^{(i)} \begin{pmatrix} b_{21} \\ b_{31} \end{pmatrix}.$$

Adding both equations together, and combining the result with Equation 2 and the definition of L_1 implies in turn that $\mathbf{s}_1 \in L_1$. Recall that by design, DEFI comes with very short secret key coefficients (which seems inevitable to ensure small public keys), as well as very short coefficients for the nonce $v_1^{(i)}$ (which this time seems inevitable to ensure small signature sizes). This qualitatively justifies the shortness of $\mathbf{s}_1$. In fact, our attack recovers $\mathbf{s}_1$ by using a lattice reduction algorithm with input the basis of L_1 described in Proposition 1.

Remark 1. Until now, we have only used a single short vector $\mathbf{s}_1$. Recall that L_1 has R-module structure, and in the case where $q = X^m + 1$, all rotations $X^i \cdot \mathbf{s}_1$ (where $\cdot$ acts on L_1 R-coordinate by R-coordinate) for $i \in [m]$ give linearly independent vectors of L_1 of equal norm. This implies two things:

- If $\mathbf{s}_1$ is unusually short, then so are its $m-1$ other rotations and lattice reduction might recover the wrong one.
- $\mathbf{s}_1$ and its rotations generate an unusually dense rank m sublattice of L_1. Instead of studying the cost of recovering $\mathbf{s}_1$ directly via lattice reduction, it makes more sense to heuristically study the reduction strength needed to recover this special dense sublattice for a given number of signatures.

The first point is in fact not really a problem, as we can simply continue the attack simultaneously with all m rotations. This is only a linear increase in complexity, so a polynomial-time attack would remain polynomial. We will explain later in Section 4.4 how we can remove remove the need for this linear increase in complexity. We cover the second point in Section 5.1.

4.2 Second Step: Recovering u_1

The aim of this Section is to adapt our first step in a way that enables us to use Equation 3 to recover the secret information $u_1^{(i)}$. The $u_2^{(i)}$ are different, so as soon as we wish to use more than one signature, an immediate lattice approach does not work. We explain how we can artificially view our problem modulo a large prime p, and how this enables us to view recovering the $u_1^{(i)}$ as yet another lattice problem. Our end result for this step is described in Observation 2 below.

Observation 2 (Informal) *If an attacker has access to a large enough number k of signatures $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(k)}$ signed with the same private key $\mathbf{B}$, then it can recover b_{41}, b_{42}, b_{43}, b_{44} and all $u_1^{(i)}$ in time polynomial in $m = \dim(R)$ and the size of the entries.*

Lemma 1. *Let $p \in \mathbb{Z}_{>0}$ be a prime number. Let $q \in \mathbb{Z}[X]$ be a monic irreducible polynomial, and $R_p := (\mathbb{Z}/p\mathbb{Z})[X]/(q)$. Then a polynomial $r \in R_p$ is invertible in R_p if and only if $\gcd(r, q) = 1$.*

In this section we fix a random large prime number p. See Section 5.2 for a discussion on the size of p.

Definition 2. *Let $k \in \mathbb{Z}_{>0}$. Assuming the vectors $(h^{(i)}, y_2^{(i)}, y_3^{(i)}, y_4^{(i)}) \in R^4$ for $i \in \mathbb{Z}_{>0}$ form a sequence of signatures obtained using the same private key, we define the following lattice*

$$L_2 = \left\{ (\alpha, \beta, \gamma, \delta, \varepsilon^{(1)}, \ldots, \varepsilon^{(k)}) \in R^{4+k} : \forall i, \varepsilon^{(i)} u_2^{(i)} \equiv_p \alpha h^{(i)} + \beta y_2^{(i)} + \gamma y_3^{(i)} + \delta y_4^{(i)} \right\}.$$

Proposition 2 (Properties of L_2). *Let $k \in \mathbb{Z}_{>0}$. The following statements hold:*

– *L_2 has rank $(k+4)m$;*
– *L_2 has ambient dimension $(k+4)m$;*
– *A basis of L_2 can be efficiently computed from the first k signatures;*
– $\mathrm{vol}(L_2) = p^{km}$.

Proof. L_2 is a q-ary lattice whose basis can be written directly in a typical NTRU-like shape:

$$\begin{pmatrix} \mathbf{0} & p\mathbf{I}_{km} \\ \mathbf{I}_{4m} & \mathbf{Y} \end{pmatrix},$$

where $\left(\mathbf{I}_{4m} \,||\, \mathbf{Y}\right)$ is derived in the same way as for L_1. Note that although from a distance, L_2 looks just like L_1 with an extra mod p condition, there is an additional subtlety in its definition: an extra multiplication by $u_2^{(i)}$ on the left of the equation defining the lattice. The modulus p was incorporated to enable us to directly write down basis vectors. Indeed by Lemma 1 all $u_2^{(i)}$ are invertible in R_p and the following vectors for $j \in [k]$ can be efficiently computed:

$$\begin{aligned}
&\left(X^j, 0, 0, 0, X^j (u_2^{(1)})^{-1} h^{(1)}, \ldots, X^j (u_2^{(k)})^{-1} h^{(k)}\right); \\
&\left(0, X^j, 0, 0, X^j (u_2^{(1)})^{-1} y_2^{(1)}, \ldots, X^j (u_2^{(k)})^{-1} y_2^{(k)}\right); \\
&\left(0, 0, X^j, 0, X^j (u_2^{(1)})^{-1} y_3^{(1)}, \ldots, X^j (u_2^{(k)})^{-1} y_3^{(k)}\right); \\
&\left(0, 0, 0, X^j, X^j (u_2^{(1)})^{-1} y_4^{(1)}, \ldots, X^j (u_2^{(k)})^{-1} y_4^{(k)}\right).
\end{aligned}$$

As in Proposition 1, these $4m$ vectors are linearly independent vectors of L_2. From the previous blockwise representation one can directly read the volume and rank, so this set of linearly independent vectors of L_2 is also generating. Of course in order to freely use the values of $u_2^{(i)}$ we assume that the first step of the attack was already executed successfully. □

We now explain the design of this lattice by exhibiting some of its short vectors. Let

$$\mathbf{s}_2' = (b_{21} + b_{31}, b_{22} + b_{32}, b_{23} + b_{33}, b_{24} + b_{34}, 1, \ldots, 1).$$

For the same reason that $\mathbf{s}_1 \in L_1$, $\mathbf{s}_2' \in L_2$. We now claim that L_2 contains another independent short vector. Let

$$\mathbf{s}_2 = (b_{41} + 1, b_{42}, b_{43}, b_{44}, u_1^{(1)}, \ldots, u_1^{(k)}).$$

For all $i \in [k]$, rewriting the last row of Equation 1 we get

$$b_{42} y_2^{(i)} + b_{43} y_3^{(i)} + b_{44} y_4^{(i)} = z_4^{(i)} - h^{(i)} b_{41}.$$

Combining this with Equation 3, for which $h = z_1$ proves that $\mathbf{s}_2 \in L_2$.

As with L_1, L_2 has R-module structure. Therefore, all shifts $X^i \cdot \mathbf{s}_2$ and $X^j \cdot \mathbf{s}_2'$ are also vectors of L_2. This lets us define the following sublattice:

$$L_2' := \langle (X^i \cdot \mathbf{s}_2)_i, (X^j \cdot \mathbf{s}_2')_j \rangle_{\mathbb{Z}} \subset L_2.$$

Experimentally, when L_2 is built using enough signatures and p is taken large enough, lattice reduction applied to L_2 happens to recover L_2' in the following way: the first $2m$ vectors of the reduced basis generate L_2' exactly. Note that L_2', by definition is independent of the chosen value of p, as neither $\mathbf{s}_2$ nor $\mathbf{s}_2'$ depend on p.

From there we have a basis for L_2', a lattice of rank $2m \in \{64, 128\}$, and it is quite clear that any lattice reduction operation on L_2' should not be too costly. Morally, this lattice is a compositum of two lattices generated by all cyclic shifts of their respective generators and we need to find a way to act on each half separately. Our goal is to recover $\mathbf{s}_2$ to get access to all the information described in Observation 2. Because $\|\mathbf{s}_2'\| < \|\mathbf{s}_2\|$, solving an SVP instance on L_2' will do nothing to help. However we notice that the first four (R)-coordinates of $\mathbf{s}_2$ should be smaller than those of $\mathbf{s}_2'$, therefore we can choose a constant value $c(k)$ and use it to skew L_2' by defining

$$L_2'' := \left\{(c(k)x_1, c(k)x_2, c(k)x_3, c(k)x_4, x_5, \ldots, x_{k+4}) : (x_i)_{i\in[k+4]} \in L_2'\right\}.$$

For a good choice of $c(k)$, *e.g.* a $c(k)$ that skews the lattice enough that the image of $\mathbf{s}_2$ in L_2'' becomes smaller in norm than the image of $\mathbf{s}_2'$, the skewed image of $\mathbf{s}_2$ in L_2'' becomes its shortest vector and can be recovered by lattice reduction in L_2''.

4.3 Final Step: Private Key Recovery

We now explain how to recover the full private key after the first two steps.

Observation 3 *If an attacker has access to $b_{2j} + b_{3j}$ and b_{4j} for $1 \leq j \leq 4$, where the b_{ij} are matrix coefficients of the private key $\mathbf{B}$ associated to $\mathbf{C}$, then it can fully recover $\mathbf{B}$ in time polynomial in $m = \dim(R)$.*

The public key is defined as $\mathbf{C} = \mathbf{B}^T\mathbf{J}\mathbf{B}$, implying the following relations on the diagonal coefficients $c_{jj} \in R$ of $\mathbf{C}$, for $1 \leq j \leq 4$:

$$c_{jj} = b_{1j}^2 + b_{2j}^2 - b_{3j}^2 - b_{4j}^2. \tag{4}$$

As would be the case after Observation 2, we now have access to all b_{4j} for $1 \leq j \leq 4$. All c_{jj} and b_{1j} are known and therefore, Equation 4 allows us to recover $b_{2j}^2 - b_{3j}^2 = (b_{2j} - b_{3j})(b_{2j} + b_{3j})$. We also know all $b_{2j} + b_{3j}$ for $1 \leq j \leq 4$ (as would be the case after Observation 1). The only remaining step is to derive $b_{2j} - b_{3j}$ from $b_{2j}^2 - b_{3j}^2$ and $b_{2j} + b_{3j}$. R being only a ring, it is impossible to simply invert $b_{2j} + b_{3j}$. However, if we pick a large enough prime p as in Section 4.2, $b_{2j} + b_{3j}$ can be inverted modulo p without any loss of information.

4.4 Exploiting the Ring Choice

In their concrete parameters and for efficiency purposes, DEFI is instantiated using $R = \mathbb{Z}[X]/(X^m + 1)$, where m is a power of two. We explain how this can be used to simplify the attack.

In Remark 1, we have seen that the first step of our attack might only recover a shift $X^j \cdot \mathbf{s}_1$ instead of our targeted $\mathbf{s}_1$. Instead of running the second step of our attack multiple times with each possible rotation, we notice that L_2 also has R-module structure, so its definition is independent of the shifts of $u_2^{(i)}$ that were obtained in the first step. Therefore the second step of the attack remains valid regardless, and it will output some $X^j \cdot \mathbf{s}_2$ that is a shift of the target secret $\mathbf{s}_2$. Now that we have handled the hardest part of the attack, we can use Equation 4 to guess and verify the correct shifts for $\mathbf{s}_1$ and $\mathbf{s}_2$. Indeed, when testing a specific pair of shifts $(X^i \cdot \mathbf{s}_1, X^j \cdot \mathbf{s}_2)$, we can use the procedure described in the proof of Observation 3 to recover a candidate value b_- for say $b_{22} - b_{32}$, using our guess b_+ for $b_{22} + b_{32}$. If this value b_- is such that $b_- + b_+$ has even and small coordinates, then the fact that a large enough prime p was chosen in the inversion implies that necessarily, the pair of shifts (i, j) has correctly been guessed. This whole procedure of guessing shifts runs at most m^2 times, which makes its runtime negligible compared to the lattice reduction steps.

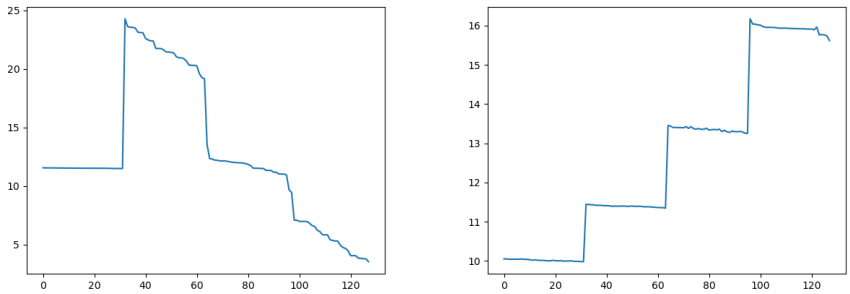
5 Some Elements to Justify the Attack

The description of the attack in Section 4 leaves a couple open questions: how can we be sure that a fast lattice reduction algorithm is enough to really recover $\mathbf{s}_1$ from our basis for L_1, as well as $\mathbf{s}_2$ from our basis for L_2? How many signatures are required to mount each step of the attack? Although most of our justification are experimental, we make a few remarks on the shape of the lattices considered. To the best of our knowledge, the behaviour of lattice reduction algorithms on lattices whose geometry resembles that of L_1 or L_2 is poorly documented, making a fully rigorous analysis near-impossible.

5.1 Analysing L_1

The first step of the attack relies on recovering $\mathbf{s}_1$ from a poor basis for L_1, a lattice which we generate using k signatures. We give will give precise experimental numbers for the minimal number of signatures k that allow us to mount this first step in Section 6. In this Section we study the shape of a reduced basis for L_1, and note that the gap between the expected norm of $\mathbf{s}_1$ and the expected norm of a shortest nonzero vector in L_1 is unusual, in the sense that it is larger than what we would expect from a typical random lattice. This in itself does not explain why polynomial-time lattice reduction algorithms seem to be enough to successfully execute this first step. The estimates from [16] do not apply, as there is no gap between $\lambda_1(L_1)$ and $\lambda_2(L_1)$. In fact, due to the presence of a sublattice of unusually small covolume and of dimension a fraction of the dimension, the situation seems to be closer to that of NTRU where it has been observed in [17,9] that when the covolume of the lattice becomes sufficiently large, lattice reduction starts recovering vectors from the dense sublattice earlier than the time we would expect it to recover short vectors. To predict lattice reduction, one might want to simulate the profile of the Gram-Schmidt norms throughout

the reduction process. This seems particularly enticing as we would expect a straight and horizontal line for the first quarter of vectors, followed by a sharp increase and then a steadily decreasing line covering the last three quarters, in the style of a Geometric Series Assumption in the presence of q-vectors. However, given that lattice reduction is so effective on L_1 (slight improvements to LLL are essentially already enough to fully reduce the basis in the cases of both DEFI-64 and DEFI-128), it is also practically impossible to simulate the behaviour of the Gram-Schmidt profile for L_1, as the transition happens so quick. If one were to generalise the scheme to larger rings and evaluate its practical security against our attack, then one would need to conduct this analysis thoroughly. In our setting however, the ring size is small enough that we can rely on experiments for this part. The Gram-Schmidt lengths pictured in Figure 2 before and after reduction remain an interesting tool to discuss the geometry of L_1. Figure 2 shows



(a) Before reduction

(b) After flatter and LLL

Fig. 2: log Gram-Schmidt norms of L_1 for $R = \mathbb{Z}[X]/(X^{32}+1)$, 10 signatures.

that the reduced basis separates into four projected sublattices with increasing covolumes. The presence of another sublattice of already short vectors on the left of the profile before reduction can only benefit reduction, in the same way as unusually short q-vectors in the input basis naturally aid reduction, as exploited in [7] and asymptotic study of [1].

In what follows we bound the gap between $\mathrm{GH}(L_1)$ and $\lambda_1(L_1)$, and show that for a large enough number k of signatures, we can expect $\mathbf{s}_1$ and its rotations to be the shortest non-zero vectors of L_1. This explains how solving an instance of SVP in L_1 allows us to recover a rotation of $\mathbf{s}_1$.

Lemma 2. *[20, Theorem 4.1.8] If* $\mathbf{A}$ *and* $\mathbf{B}$ *are nonnegative Hermitian square matrices in* $M_n(\mathbb{C})$*, then*

$$\det(\mathbf{A}+\mathbf{B})^{1/n} \geq \det(\mathbf{A})^{1/n} + \det(\mathbf{A})^{1/n}.$$

Proposition 3 (Volume of L_1). *Let $k \in \mathbb{Z}_{>0}$ be an integer divisible by 4. If we write the computed basis for L_1 blockwise as*

$$\left(\mathbf{I}_{4m} \mid\mid \mathbf{A}_1 \mid\mid \ldots \mid\mid \mathbf{A}_{k/4}\right)$$

with square matrices $\mathbf{A}_i$, then

$$\mathrm{vol}(L_1) \geq \left(\frac{k}{4}\right)^{2m} \left(\min_{1 \leq i \leq k} \det(\mathbf{A}_i)\right).$$

If we also assume that the $\mathbf{A}_i$ are independent random variables that follow the same distribution, then

$$\mathbb{E}\left(\mathrm{vol}(L_1)^{\frac{1}{4m}}\right) \geq \frac{\sqrt{k}}{2} \sqrt{\mathbb{E}(\det(\mathbf{A}_1)^{\frac{2}{4m}})}.$$

Proof. We prove this identity using Lemma 2, inductively on k. Indeed,

$$\begin{aligned}
\mathrm{vol}(L_1)^{\frac{2}{4m}} &= \det\left(\left(\mathbf{I}_{4m} \mid\mid \mathbf{A}_1 \mid\mid \ldots \mid\mid \mathbf{A}_{k/4}\right) \cdot \left(\mathbf{I}_{4m} \mid\mid \mathbf{A}_1 \mid\mid \ldots \mid\mid \mathbf{A}_{k/4}\right)^T\right)^{\frac{1}{4m}} \\
&= \det\left(\mathbf{I}_{4m} + \mathbf{A}_1\mathbf{A}_1^T + \ldots + \mathbf{A}_{k/4}\mathbf{A}_{k/4}^T\right)^{\frac{1}{4m}} \\
&\geq 1 + \sum_{i=1}^{k/4} \det\left(\mathbf{A}_i\mathbf{A}_i^T\right)^{\frac{1}{4m}} \\
&\geq \frac{k}{4} \min_{1 \leq i \leq k} \det(\mathbf{A}_i)^{\frac{2}{4m}},
\end{aligned}$$

where we used the fact that the $\mathbf{A}_i\mathbf{A}_i^T$ are all nonnegative, Hermitian and square. The second identity immediately follows from linearity of expectation. □

The following Proposition explains the behaviour of the gap for L_1 generated from asymptotically many signatures.

Proposition 4. *Let $k \in \mathbb{Z}_{>0}$ be an integer divisible by 4. Using the same notations as in Proposition 3, and assuming the $\mathbf{A}_i$ are independent random variable that follow the same distribution, then*

$$\frac{\mathrm{GH}(L_1)}{\lambda_1(L_1)} \geq (1 + o_{k,m}(1)) \sqrt{\frac{m}{32\pi e \lambda_u}} \min \det(\mathbf{A}_i)^{\frac{1}{4m}},$$

and

$$\mathbb{E}\left(\frac{\mathrm{GH}(L_1)}{\|\mathbf{s}_1\|}\right) \leq \sqrt{\frac{m}{4\pi e \lambda_u}} \sqrt{\mathbb{E}(\det(\mathbf{A}_1)^{\frac{2}{4m}})}.$$

Proof. We now show that $\mathbf{s}_1$ is somewhat short. Indeed, the coefficients of the secret key $\mathbf{B}$ are themselves bounded, and the $u_2^{(i)}$ are generated according to a

distribution that has short expected norm. More precisely, $u_2^{(i)}$ consists of exactly λ_u coordinates in $\{\pm 2, \pm 4\}$, which means that we have

$$\begin{aligned}\|\mathbf{s}_1\|^2 &\leq \sum_{j=1}^{4} \|b_{2j} + b_{3j}\|^2 + \sum_{i=1}^{k} \|u_2^{(i)}\|^2 \\ &\leq 16(\delta_{\mathbf{B}_{22}} + \lambda_u k).\end{aligned}$$

The gap between the Gaussian heuristic on L_1 and the norm of the shortest vector can therefore be bounded using Proposition 3, where the $o_k(\cdot)$ is taken as $k \to \infty$:

$$\begin{aligned}\frac{\mathrm{GH}(L_1)}{\lambda_1(L_1)} \geq \frac{\mathrm{GH}(L_1)}{\|\mathbf{s}_1\|} &\geq (1 + o_m(1)) \frac{\sqrt{\frac{4m}{2\pi e}} \mathrm{vol}(L_1)^{\frac{1}{4m}}}{\sqrt{16(\delta_{\mathbf{B}_{22}} + \lambda_u k)}} \\ &\geq (1 + o_m(1)) \frac{\sqrt{\frac{4m}{2\pi e}} \sqrt{k/4} \min \det(\mathbf{A}_i)^{\frac{1}{4m}}}{\sqrt{16(\delta_{\mathbf{B}_{22}} + \lambda_u k)}} \\ &= (1 + o_{k,m}(1)) \sqrt{\frac{m}{32\pi e \lambda_u}} \min \det(\mathbf{A}_i)^{\frac{1}{4m}}.\end{aligned}$$

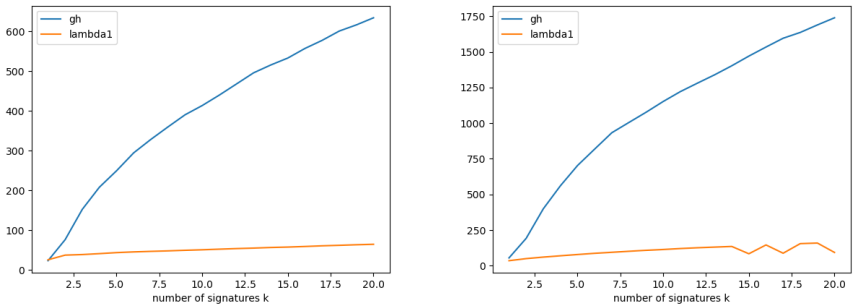
Using the second item of Proposition 3, and the inequality $\|\mathbf{s}_1\| \geq \sqrt{2k\lambda_u}$ we can now bound

$$\begin{aligned}\mathbb{E}\left(\frac{\mathrm{GH}(L_1)}{\|\mathbf{s}_1\|}\right) &\leq \sqrt{\frac{4m}{2\pi e}} \frac{\sqrt{k/4}}{\sqrt{2k\lambda_u}} \sqrt{\mathbb{E}(\det(\mathbf{A}_1)^{\frac{2}{4m}})} \\ &\leq \sqrt{\frac{m}{4\pi e \lambda_u}} \sqrt{\mathbb{E}(\det(\mathbf{A}_1)^{\frac{2}{4m}})}.\end{aligned}$$

□

Proposition 4 helps justify two things. First, it is well-known that increasing the gap between a vector's length and the Gaussian heuristic while leaving the covolume to a fixed value equates to increasing the Hermite factor, which heuristically leads to an easier lattice problem. Therefore, as this gap increases with k, it makes sense that more signatures lead to an easier lattice problem. Second, a larger gap between $GH(L_1)$ and $\|\mathbf{s}_1\|$ means that it is more likely (the probability should in fact be overwhelming as in [19, Theorem 6]) that $\mathbf{s}_1$ and its rotations are the true shortest vectors in L_1. The dimension of the lattice is 128 for DEFI-64 and 256 for DEFI-128, making running simple lattice reduction algorithms feasible. Figure 3 compares the Gaussian heuristic for L_1 with the size of the shortest vector recovered by a run of flatter and LLL in the case of DEFI-64 and DEFI-128.

Although both curves seem to diverge fast, their main terms are both up to a constant equivalent to $\sqrt{k}$, therefore their ratio converges towards a constant value. Asymptotically, this gap is less than the gap that one would observe in the NTRU lattice [3].



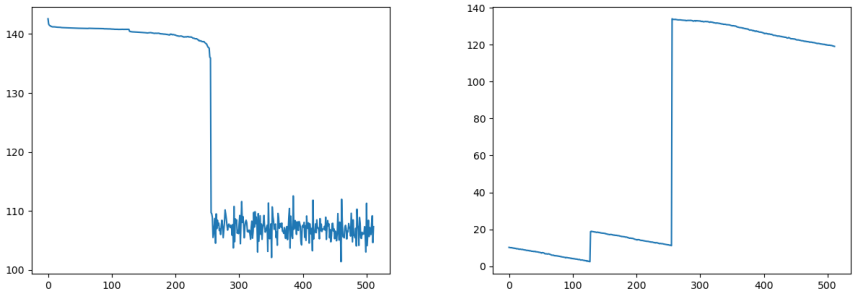
(a) DEFI-64 (b) DEFI-128

Fig. 3: Comparing the size of $\mathbf{s}_1$ with the Gaussian heuristic in L_1, for increasing number of signatures k.

5.2 Analysing L_2

Once the first step has successfully been executed, the second step selects a prime modulus p, and then generates a lattice L_2 with the data from k signatures. Contrarily to what happened with L_1, where the number of signatures did not influence the dimension of the lattice, here $\dim(L_2) = (k+4)m$, and therefore as we are going to have to reduce L_2, it is of the utmost importance to limit the number of signatures included into L_2 to the minimum possible.

We start by observing the Gram-Schmidt profiles for L_2, before and after reduction by LLL.



(a) Before reduction (b) After LLL

Fig. 4: log Gram-Schmidt norms of L_2 for $R = \mathbb{Z}[X]/(X^{64}+1)$, 4 signatures

Figure 4 confirms the observations from Section 4.2: if the modulus p is large enough, lattice reduction separates the vectors in L_2 that live in a sublattice independent of p. We notice a first sublattice of dimension $4m$, as well as another of dimension $2m$, which in fact corresponds exactly to the lattice L_2' defined in Section 4.2.

To prove that LLL or a stronger form of lattice reduction recovers L_2' would require answering a lot of questions combining sublattice finding with lattice reduction theory. Although some algorithms for the densest sublattice problem [6] have been proposed, or studied in the very particular case of NTRU [9], the practical solution to the following question is yet not well understood: given a rank n lattice with the promise that it has an unusually dense sublattice of rank n', how hard is it to recover such a sublattice? This question generalises the more common study of lattice reduction algorithms to solve SVP in the presence of an unusually short vector (or in other words an unusually dense rank 1 sublattice). It is possible to formalise the wording *unusually dense* by comparing the covolume of the sublattice with the appropriate value of the expected covolume of a random rank n' sublattice, as described in [27], but we consider this to be outside of the scope of our account of the proposed attack on DEFI, and choose parameters (number of signatures, size of p, lattice reduction algorithms) that consistently and efficiently recover the sublattice L_2'.

5.3 Analysing the Key-Recovery Step

If the first two steps have been performed successfully, then it is possible to prove Observation 3.

Lemma 3. *Let $a, b, c \in R$ be ring elements such that $a = bc$, $\gcd(b, q) = 1$. Let p be a prime number such that $p \geq 2\|c\|_\infty$. If for $x \in R$, $\tilde{x}$ denotes the class of x in $R_p = (\mathbb{Z}/p\mathbb{Z})[X]/(q)$, and $\mathrm{Round}(\tilde{x})$ is the representative of $\tilde{x}$ in R that has minimal ∞-norm, then*

$$c = \mathrm{Round}(\tilde{b}^{-1}\tilde{a}).$$

Proof. First, $\tilde{b}$ is invertible in R_p thanks to Lemma 1. Therefore, $\tilde{b}^{-1}\tilde{a} = \tilde{c}$ in R_p. Now $\|\mathrm{Round}(\tilde{c})\|_\infty \leq \frac{p}{2}$ and there is only one such representative of $\tilde{c}$. The bound on $\|c\|_\infty$ implies that $c = \mathrm{Round}(\tilde{c})$. □

Proof (Of Observation 3). As mentioned above, we assume that we can apply Observations 1 and 2 to recover the sums $b_{2j} + b_{3j}$ and the b_{4j} for $1 \leq j \leq 4$. Then using Lemma 3 onto Equation 4 we obtain that if $p \geq 2\|b_{2j} - b_{3j}\|_\infty$ is a prime, the smallest representative of $(b_{2j} \tilde{+} b_{3j})^{-1}(\tilde{c_{jj}} - \tilde{b}_{1j}^2 + \tilde{b}_{4j}^2)$ in R is $b_{2j} - b_{3j}$. The half-sums and half-differences give us all remaining coefficients of the secret key $\mathbf{B}$. Finally, note that $2\|b_{2j} - b_{3j}\|_\infty \leq 4\max(\delta_{\mathbf{B}_{21}}, \delta_{\mathbf{B}_{22}})$, so any prime p larger than this value will suffice. □

6 Experiments

We ran all experiments on a personal laptop with the following processor information: `intel i7-1065G7 CPU@1.3GHz`. Our code for the attack is available online at

https://gitlab.inria.fr/hbambury/defi-nitely-broken.

The full secret-key recovery attack for the challenge instance DEFI-64 runs in less than 20 seconds with 3 signatures. For reference, we give the solution to the challenge in the appendix. We now focus only on the strongest security parameters proposed in [13], DEFI-128.

6.1 Running the Attack

Using the code available at [11], we generated 100 DEFI-128 public keys with corresponding signatures, and tested our attack 100 times.

First step. Using the flatter software [24] followed by fplll's LLL implementation [26] as our main lattice reduction tools for reducing L_1 in the first step, we are able to recover the nonces u_2 in less than 50 seconds on average. In all 100 instances, the first step failed with 8 signatures, but was successful with 9. If one is willing to pay the cost of time and run stronger lattice reduction algorithms, then it is likely than one can reduce the number of required signatures. We chose not to explore this path.

Second step. Using flatter combined with LLL on L_2 with 4 signatures and a random 100-bit prime number for p recovered L_2' in all 100 instances, with an average runtime under 180 seconds. 3 signatures were not enough to separate the $2m$-dimensional sublattice from the $4m$-dimensional one (see Figure 4b). We were able to recover the nonces u_1 using fplll's implementation of BKZ with blocksize 20 on L_2'' in all 100 instances, with an average runtime under 30 seconds.

Key-recovery. The runtime for the last step is negligible compared to the first two step, and this step is guaranteed to work. We conclude that our attack was successfully able to recover the private key in all 100 of our DEFI-128 challenges using 9 signatures, and in under 5 minutes for each.

6.2 Minor Improvements

Lattice weights. It comes as no surprise that the sizes of the coordinates of the short vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ can be roughly predicted from the parameters of the scheme. Indeed, they are directly tied to: on one side the generation of the private key $\mathbf{B}$, and on the other the generation of the nonces u_1 and u_2. In practice, we add some weights to the different columns of the lattices we reduce to ensure that the target vector has balanced coordinates. Even if done very roughly, this allowed us to lower the number of required signatures without the need for stronger lattice reduction.

On the use of flatter. The most expensive part of our attack is by far the lattice reduction step. Lattice reduction can become costly when the lattice dimension is large, or when the size of the input lattice vectors grows. The vectors here are of very reasonable sizes, but the lattices grow in dimension. L_1 is always 256-dimensional, and L_2 with 4 signatures is 512-dimensional, which is far too big for any naïve implementation of LLL. The algorithm of [24] enables us to deal with such high dimensions in only a few minutes, we use it as a pre-processing step for LLL[3].

Even lattice intersection. In Section 4.1, we aim at recovering a short vector $\mathbf{s}_1$ that contains the elements $u_2^{(i)}$. Because of the trapdoor construction described in Algorithm 1, $u_2^{(i)}$ consists of λ_u coefficients that are all in $\{\pm 2, \pm 4\}$. Consequently,

$$\mathbf{s}_1 \in L_1 \cap L_{even},$$

where $L_{even} \cong \mathbb{Z}^{4m} \times (2\mathbb{Z})^{km}$ is the lattice whose last km coordinates are all even. It might seem natural that using this extra information on the shape of the nonces should help us, especially as this lattice intersection can be efficiently computed through duality. We do not observe any substantial experimental improvement when considering this intersection.

Conclusion: Discussion and Perspectives

We have presented a full key-recovery attack on all proposed parameters of [13], a signature scheme based on an innovative problem involving isotropic vectors of non-definite quadratic forms. Our attack is well motivated, and was shown to work experimentally on every challenge instance we generated.

Sublattice recovery. A point that remained unclear to us in the analysis of our attack is the analysis of the sublattice recovery problem. We show an example of a situation where it would be interesting to understand how, why and when lattice reduction recovers a given unusually dense sublattice. This question having ties with the study of NTRU lattices, we believe it might be of independent interest.

Fixing DEFI? We see no obvious countermeasure to our attack, other than increasing parameters or radically changing the procedure for generating an isotropic vector. A direct fix by adapting parameter values would require a careful study of our lattice attack and ensure that more computationally intense lattice reduction as allowed by the desired security requirements does not lead to any exploitable leaks, even when many signatures are available to the attacker.

[3] Using both flatter and an LLL implementation might sound somewhat redundant. Flatter is significantly faster than LLL for lattices in large dimensions, but does not guarantee an LLL-reduced basis. We found that running LLL after flatter improved the basis quality for a minimal overhead.

Even if this were possible, it would most certainly increase the parameters for the scheme, making it less competitive, and would still not consist in a sound security proof. A true fix would require changing the procedure for generating an isotropic vector in such a way that the output distribution would become independent of the secret key.

An interesting new assumption. The idea behind the scheme remains new and interesting. The claimed hard problem of recovering a unimodular matrix $\mathbf{B}$ from the public key $\mathbf{C} = \mathbf{B}^T\mathbf{J}\mathbf{B}$ certainly looks a lot like the *Lattice Isomorphism Problem* [10,2], only that it is defined here with $\mathbf{J}$, which is not positive definite, making it different from the classical lattice problem. This Isotropic Quadratic Form problem is not well studied from the cryptographic point of view and would benefit from more constructions and direct cryptanalysis using algorithmic ideas from the study of reduction of quadratic forms, as well as mathematical ideas on the classification or decomposition of isotropic forms.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 885394).

References

1. Bambury, H., Nguyen, P.Q.: Improved provable reduction of NTRU and hypercubic lattices. In: Saarinen, M.J., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I. pp. 343–370. Springer, Cham, Switzerland, Oxford, UK (June 12–14, 2024). https://doi.org/10.1007/978-3-031-62743-9_12
2. Bennett, H., Ganju, A., Peetathawatchai, P., Stephens-Davidowitz, N.: Just how hard are rotations of $\mathbb{Z}^n$? algorithms and cryptography with the simplest lattice. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part V. Lecture Notes in Computer Science, vol. 14008, pp. 252–281. Springer, Cham, Switzerland, Lyon, France (April 23–27, 2023). https://doi.org/10.1007/978-3-031-30589-4_9
3. Chen, C., Danba, O., Hoffstein, J., Hulsing, A., Rijneveld, J., Schanck, J.M., Saito, T., Schwabe, P., Whyte, W., Xagawa, K., Yamakawa, T., Zhang, Z.: Ntru algorithm specifications and supporting documentation (9 2020)
4. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 1–20. Springer Berlin Heidelberg, Germany, Seoul, South Korea (December 4–8, 2011). https://doi.org/10.1007/978-3-642-25385-0_1
5. Cloudflare: A look at the latest post-quantum signature standardization candidates. `https://blog.cloudflare.com/another-look-at-pq-signatures/` (November 7 2024)

6. Dadush, D., Micciancio, D.: Algorithms for the densest sub-lattice problem. In: Khanna, S. (ed.) 24th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1103–1122. ACM-SIAM, New Orleans, LA, USA (January 6–8, 2013). https://doi.org/10.1137/1.9781611973105.79
7. Ducas, L., Espitau, T., Postlethwaite, E.W.: Finding short integer solutions when the modulus is small. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2023, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 150–176. Springer, Cham, Switzerland, Santa Barbara, CA, USA (August 20–24, 2023). https://doi.org/10.1007/978-3-031-38548-3_6
8. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium – submission to round 2 of the nist post-quantum project (March 2019)
9. Ducas, L., van Woerden, W.P.J.: NTRU fatigue: How stretched is overstretched? In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021, Part IV. Lecture Notes in Computer Science, vol. 13093, pp. 3–32. Springer, Cham, Switzerland, Daejeon, South Korea (December 7–11, 2021). https://doi.org/10.1007/978-3-030-92068-5_1
10. Ducas, L., van Woerden, W.P.J.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part III. Lecture Notes in Computer Science, vol. 13277, pp. 643–673. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – June 3, 2022). https://doi.org/10.1007/978-3-031-07082-2_23
11. Feussner, M.: Defi reference implementation. `https://github.com/martinfeussner/DEFI/tree/dd038b3/DEFI128` (May 3, 2024)
12. Feussner, M., Semaev, I.: Digital signature algorithms EHTv3 and EHTv4 – submission to round 1 of the nist additional call for post-quantum signatures (2023)
13. Feussner, M., Semaev, I.: Isotropic quadratic forms, diophantine equations and digital signatures. Cryptology ePrint Archive, Report 2024/679, version 1 (2024), `https://eprint.iacr.org/archive/2024/679/20240503:175841`, announced on May 6th, 2024 on the NIST-pqc mailing list.
14. Feussner, M., Semaev, I.: Isotropic quadratic forms, diophantine equations and digital signatures, DEFIv2. Cryptology ePrint Archive, Report 2024/679, version 2 (2024), `https://eprint.iacr.org/archive/2024/679/20241105:105112`, last updated November 2024.
15. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU (March 2019)
16. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 31–51. Springer Berlin Heidelberg, Germany, Istanbul, Turkey (April 13–17, 2008). https://doi.org/10.1007/978-3-540-78967-3_3
17. Kirchner, P., Fouque, P.A.: Revisiting lattice attacks on overstretched NTRU parameters. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10210, pp. 3–26. Springer, Cham, Switzerland, Paris, France (April 30 – May 4, 2017). https://doi.org/10.1007/978-3-319-56620-7_1
18. Lenstra, A.K., Lenstra, Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Ann. **261**, 513–534 (1982)

19. Li, J., Nguyen, P.Q.: A complete analysis of the bkz lattice reduction algorithm. Journal of Cryptology **38** (December 13, 2024). https://doi.org/10.1007/s00145-024-09527-0
20. Marcus, M., Minc, H.: A survey of matrix theory and matrix inequalities. Dover, New York (1992)
21. NIST: Call for additional digital signature schemes for the post-quantum cryptography standardization process (2022), `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf`
22. NIST: Post-Quantum Cryptography Standardization (2022), `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization`
23. PQShield: Post-quantum signatures zoo. `https://pqshield.github.io/nist-sigs-zoo/` (October 28 2024)
24. Ryan, K., Heninger, N.: Fast practical lattice reduction through iterated compression. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part III. pp. 3–36. Lecture Notes in Computer Science, Springer, Cham, Switzerland, Santa Barbara, CA, USA (August 20–24, 2023). https://doi.org/10.1007/978-3-031-38548-3_1
25. Schnorr, C.P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Programming **66**, 181–199 (1994)
26. The FPLLL development team: fpylll, a Python wrapper for the fplll lattice reduction library, Version: 0.6.1 (2024), `https://github.com/fplll/fpylll`, available at `https://github.com/fplll/fpylll`
27. Thunder, J.L.: Higher-dimensional analogs of hermite's constant. Michigan Mathematical Journal **45**, 301–314 (1998)

Appendix A Solution to the DEFI-64 Challenge

Up to sign, writing $\mathbf{B} = (b_{i,j})_{i,j\in[4]}$ as elements of $\mathbb{Z}[X]/(X^{32}+1)$:

$$
\begin{aligned}
b_{1,1} =& 1\\
b_{1,2} =& 0\\
b_{1,3} =& 0\\
b_{1,4} =& 0\\
b_{2,1} =& -2X^{31} + X^{30} - 2X^{28} - X^{15} - 2X^{12} + 2X^{10} - 2X^8 + 2X^3 + 2X\\
b_{2,2} =& X^{31} - X^{30} + 3X^{28} + X^{27} + X^{22} + X^{21} + X^{19} + 2X^{18} + X^{16} + X^{15}\\
&+ 3X^{13} + X^{12} - X^{11} + X^9 + X^7 - X^5 + 2X^4 + X^3 - X^2 + X\\
b_{2,3} =& 2X^{31} + 2X^{30} - 2X^{27} + X^{26} + 3X^{25} - X^{24} - 2X^{23} - X^{21} + X^{20} - 2X^{18} + 5X^{16} + X^{15} + X^{14}\\
&- 2X^{13} - X^{12} - X^{11} + 3X^{10} - X^9 - X^8 + 2X^7 + 2X^5 + X^4 - X^3 - 2X^2 + 4X - 2\\
b_{2,4} =& X^{28} - X^{27} + X^{26} + X^{22} + X^{21} + X^{20} + X^{19} - X^{18} + X^{17} + X^{16} + 2X^{14}\\
&+ X^{13} + 2X^{12} + X^{11} + X^{10} - X^9 + X^6 + X^5 + X - 1\\
b_{3,1} =& -2X^{29} - 2X^{26} + X^{24} + X^{18} - 2X^{16} - 2X^{14} - 2X^5 - X^2 - X\\
b_{3,2} =& X^{31} - 2X^{30} - 2X^{29} + 2X^{28} - X^{26} - 3X^{24} - X^{23} + 2X^{22} - X^{20} + 2X^{16} - X^{15}\\
&- X^{14} + 4X^{13} - X^{11} + X^{10} - X^9 + 3X^7 + X^6 - X^5 + X^3 + 2X + 1\\
b_{3,3} =& 2X^{30} + 3X^{29} + X^{28} - 6X^{27} + 2X^{25} + 2X^{24} - X^{23} - X^{22} - 5X^{21} + X^{20} + X^{19} - 4X^{18} + X^{16}\\
&+ 3X^{14} + X^{13} - 4X^{12} + X^{11} + 5X^{10} + 2X^9 - X^8 - 2X^6 + 2X^5 + 5X^4 - 2X^3 - 4X^2 + 3X - 2\\
b_{3,4} =& -X^{31} - X^{30} + X^{29} - 2X^{27} + X^{26} - 2X^{25} - X^{24} - 2X^{23} - X^{22} + X^{20} - X^{18}\\
&- X^{17} + 2X^{14} + X^{13} + 2X^{11} + X^{10} + X^7 + X^5 - X^3 + X\\
b_{4,1} =& -X^{30} + 2X^{20} + 2X^{11} + 2X^{10} - 2X^7 + 2X^4 - 2X^3 - X^2 - 1\\
b_{4,2} =& 3X^{29} + X^{28} - X^{27} + X^{26} - X^{24} + X^{23} - 3X^{21} + X^{18} + X^{15} - X^{14} - X^{13}\\
&- X^{12} - 2X^{11} - X^9 - X^8 - X^7 - 3X^6 - X^5 + X^4 - X^2 - 1\\
b_{4,3} =& X^{31} + 2X^{30} + X^{29} - X^{28} + 4X^{26} - 3X^{25} - 3X^{24} - 3X^{23} + X^{22} + X^{20} - X^{18} + 2X^{17} + 3X^{16}\\
&- X^{15} + X^{14} - X^{13} + 2X^{12} + X^{11} - X^{10} - 6X^9 + 3X^6 + X^5 + X^4 - X^3 + X^2 - 3\\
b_{4,4} =& 2X^{31} + 2X^{30} + X^{28} + X^{26} - X^{25} + X^{23} - X^{21} - X^{19} + X^{16} - X^{15}\\
&- X^{13} + X^{11} - X^{10} - X^9 - 2X^7 - X^6 - 2X^5 - X^4 - X^3 + X^2 - X.
\end{aligned}
$$