Uprooting the Falcon Tree?

How to Recover Secret Keys from Gram–Schmidt Norms

Pierre-Alain Fouque[1,2], Paul Kirchner[1,2], Mehdi Tibouchi[3], Alexandre Wallet[3], and Yang Yu[1,2]

[1] IRISA, France
paul.kirchner@irisa.fr
[2] Univ Rennes, France
{pa.fouque,yang.yu0986}@gmail.com
[3] NTT Corporation, Japan
{mehdi.tibouchi.br,alexandre.wallet.th}@hco.ntt.co.jp

Abstract. In this paper, we initiate the study of side-channel leakage in hash-and-sign lattice-based signatures, with particular emphasis on the two efficient implementations of the original GPV lattice-trapdoor paradigm for signatures, namely NIST second-round candidate FALCON and its simpler predecessor DLP. Both of these schemes implement the GPV signature scheme over NTRU lattices, achieving great speed-ups over the general lattice case. Our results are mainly threefold.

First, we identify a specific source of side-channel leakage in most implementations of those schemes. Signing in lattice-based hash-and-sign schemes involves sampling a lattice point according to a Gaussian distribution. This reduces to sampling several one-dimensional discrete Gaussian distributions with standard deviations determined by the Gram–Schmidt norms of the secret lattice basis. Our observation is that those norms often leak through timing side-channels in the implementation of the one-dimensional Gaussian samplers.

Second, we elucidate the link between this leakage and the secret key, by showing that the entire secret key can be efficiently reconstructed solely from those Gram–Schmidt norms. The result makes heavy use of the algebraic structure of the corresponding schemes, which work over a power-of-two cyclotomic field. To establish it, we propose efficient algorithms of independent interest which, given the leading principal minors of the matrix associated to a totally positive field element (in the power basis for DLP and the bit-reversed order basis for FALCON) recover the element up to conjugation. In the case of those schemes, that element is $f\bar{f} + g\bar{g}$, where (f, g) is the NTRU-style secret key. We then show that this element combined with the verification key suffices to recover the entire secret efficiently.

Third, we concretely demonstrate the side-channel attack against DLP. The challenge is that timing information only provides an approximation of the Gram–Schmidt norms (with an accuracy increasing with the number of traces), and our algebraic recovery technique needs to be combined with pruned tree search in order to apply it to approximated values. Experimentally, we show that around 2^{35} DLP traces are enough to reconstruct the entire key with good probability. Carrying out a similar experiment against FALCON is left as an open problem, however, since the recursive nature of our bit-reversed order recovery algorithm does not accommodate approximate inputs easily. Nevertheless, our results do underscore the importance of constant time implementations particularly for schemes using Gaussian sampling.

Keywords: Cryptanalysis · Lattice-Based Cryptography · NTRU · Lattice Gaussian Sampling · Timing Attacks · Algebraic Number Theory

1 Introduction

Lattice-based signatures. Lattice-based cryptography has proved to be a versatile way of achieving a very wide range of cryptographic primitives with strong security guarantees that are also believed to hold in the postquantum setting. For a while, it was largely confined to the realm of theoretical cryptography, mostly concerned with asymptotic efficiency, but it has made major strides towards practicality in recent years. Significant progress has been made in terms of practical constructions, refined concrete security estimates and fast implementations. As a result, lattice-based schemes are seen as strong contenders in the NIST postquantum standardization process.

In terms of practical *signature schemes* in particular, lattice-based constructions broadly fit within either of two large frameworks: Fiat–Shamir type constructions on the one hand, and hash-and-sign constructions on the other.

Fiat–Shamir lattice based signatures rely on a variant of the Fiat–Shamir paradigm [18] developed by Lyubashevsky, called "Fiat–Shamir with aborts" [32], which has proved particularly fruitful. It has given rise to numerous practically efficient schemes [24, 11, 2] including the two second round NIST candidates Dilithium [12, 34] and qTESLA [6].

The hash-and-sign family has a longer history, dating back to Goldreich–Goldwasser–Halevi [23] signatures as well as NTRUSign [25]. Those early proposals were shown to be insecure [20, 22, 41, 14], however, due to a statistical dependence between the distribution of signatures and the signing key. That issue was only overcome with the development of lattice trapdoors by Gentry, Peikert and Vaikuntanathan [21]. In the GPV scheme, signatures follow a distribution that is provably independent of the secret key (a discrete Gaussian supported on the public lattice), but which is hard to sample from without knowing a secret, short basis of the lattice. The scheme is quite attractive from a theoretical standpoint (for example, it is easier to establish QROM security for it than for Fiat–Shamir type schemes), but suffers from large keys and a potentially costly procedure for discrete Gaussian sampling over a lattice. Several follow-up works have then striven to improve its concrete efficiency [43, 50, 38, 35, 15], culminating in two main efficient and compact implementations: the scheme of Ducas, Lyubashevsky and Prest (DLP) [13], and its successor, NIST second round candidate FALCON [48], both instantiated over NTRU lattices [25] in power-of-two cyclotomic fields. One can also mention NIST first round candidates pqNTRUSign [53] and DRS [45] as members of this family, the latter of which actually fell prey to a clever statistical attack [52] in the spirit of those against GGH and NTRUSign.

Side-channel analysis of lattice-based signatures. With the NIST postquantum standardization process underway, it is crucial to investigate the security of lattice-based schemes not only in a pure algorithmic sense, but also with respect to implementation attacks, such as side-channels. For lattice-based signatures constructed using the Fiat–Shamir paradigm, this problem has received a significant amount of attention in the literature, with numerous works [10, 44, 16, 7, 5, 51] pointing out vulnerabilities with respect to timing attacks, cache attacks, power analysis and other types of side-channels. Those attacks have proved particularly devastating against schemes using discrete Gaussian sampling, such as the celebrated BLISS signature scheme [11]. In response, several countermeasures have also been proposed [29, 40, 28], some of them provably secure [4, 5], but the side-channel arms race does not appear to have subsided quite yet.

In contrast, the case of hash-and-sign lattice-based signatures, including DLP and FALCON, remains largely unexplored, despite concerns being raised regarding their vulnerability to side-channel attacks. For example, the NIST status report on first round candidates, announcing the

selection of FALCON to the second round, notes that "more work is needed to ensure that the signing algorithm is secure against side-channel attacks". The relative lack of cryptanalytic works regarding these schemes can probably be attributed to the fact that the relationship between secret keys and the information that leaks through side-channels is a lot more subtle than in the Fiat–Shamir setting.

Indeed, in Fiat–Shamir style schemes, the signing algorithm uses the secret key very directly (it is combined linearly with other elements to form the signature), and as a result, side-channel leakage on sensitive variables, like the random nonce, easily leads to key exposure. By comparison, the way the signing key is used in GPV type schemes is much less straightforward. The key is used to construct the trapdoor information used for the lattice discrete Gaussian sampler; in the case of the samplers [31, 21, 15] used in GPV, DLP and FALCON, that information is essentially the Gram–Schmidt orthogonalization (GSO) of a matrix associated with the secret key. Moreover, due to the way that GSO matrix is used in the sampling algorithm, only a small amount of information about it is liable to leak through side-channels, and how that small amount relates to the signing key is far from clear. To the best of our knowledge, neither the problem of identifying a clear side-channel leakage, nor that of relating that such a leakage to the signing key have been tackled in the literature so far.

Our contributions. In this work, we initiate the study of how side-channel leakage impacts the security of hash-and-sign lattice-based signature, focusing our attention to the two most notable practical schemes in that family, namely DLP and FALCON. Our contributions towards that goal are mainly threefold.

First, we identify a specific leakage of the implementations of both DLP and FALCON (at least in its original incarnation) with respect to timing side-channels. As noted above, the lattice discrete Gaussian sampler used in signature generation relies on the Gram–Schmidt orthogonalization of a certain matrix associated with the secret key. Furthermore, the problem of sampling a discrete Gaussian distribution supported over the lattice is reduced to sampling one-dimensional discrete Gaussians with standard deviations computed from the norms of the rows of that GSO matrix. In particular, the one-dimensional sampler has to support varying standard deviations, which is not easy to do in constant time. Unsurprisingly, the target implementations both leak that standard deviation through timing side-channels; specifically, they rely on rejection sampling, and the acceptance rate of the corresponding loop is directly related to the standard deviation. As a result, timing attacks will reveal the Gram–Schmidt norms of the matrix associated to the secret key (or rather, an approximation thereof, to a precision increasing with the number of available samples).

Second, we use algebraic number theoretic techniques to elucidate the link between those Gram–Schmidt norms and the secret key. In fact, we show that the secret key can be entirely reconstructed from the knowledge of those Gram–Schmidt norms (at least if they are known *exactly*), in a way which crucially relies on the algebraic structure of the corresponding lattices.

Since both DLP and FALCON work in an NTRU lattice, the signing key can be expressed as a pair (f, g) of small elements in a cyclotomic ring $\mathcal{R} = \mathbb{Z}[\zeta]$ (of power-of-two conductor, in the case of those schemes). The secret, short basis of the NTRU lattice is constructed by blocks from the multiplication matrices of f and g (and related elements F, G) in a certain basis of $\mathcal{R}$ as a $\mathbb{Z}$-algebra (DLP uses the usual power basis, whereas FALCON uses the power basis in *bit-reversed order*; this apparently small difference interestingly plays a crucial role in this work). It is then easily seen that the Gram matrix of the first half of the lattice basis is essentially the multiplication matrix associated with the element $u = f\bar{f} + g\bar{g}$, where the bar denotes the complex conjugation

$\bar{\zeta} = \zeta^{-1}$. From that observation, we deduce that knowing the Gram–Schmidt norms of lattice basis is essentially equivalent to knowing the leading principal minors of the multiplication matrix of u, which is a real, totally positive element of $\mathcal{R}$.

We then give general efficient algorithms, both for the power basis (DLP case) and for the bit-reversed order power basis (FALCON case), which recover an arbitrary totally positive element u (up to a possible automorphism of the ambient field) given the leading principal minors of its multiplication matrix. The case of the power basis is relatively easy: we can actually recover the coefficients iteratively one by one, with each coefficient given as a solution of quadratic equation over $\mathbb{Q}$ depending only on the minors and the previous coefficients. The bit-reversed order power basis is more contrived, however; recovery is then carried out recursively, by reduction to the successive subfields of the power-of-two cyclotomic tower.

Finally, to complete the recovery, we need to deduce f and g from u. We show that this can be done using the public key $h = g/f \bmod q$: we can use it to reconstruct both the relative norm $f\bar{f}$ of f, and the ideal $(f) \subset \mathcal{R}$. That data can then be plugged into the Gentry–Szydlo algorithm [22] to obtain f in polynomial time, and hence g. Those steps, though simple, are also of independent interest, since they can be applied to the side-channel attack against BLISS described in [16], in order to get rid of the expensive factorization of an algebraic norm, and hence make the attack efficient for all keys (instead of a small percentage of weak keys as originally stated).

Our third contribution is to actually collect timing traces for the DLP scheme and mount the concrete key recovery. This is not an immediate consequence of the previous points, since our totally positive element recovery algorithm a priori requires the *exact* knowledge of Gram–Schmidt norms, whereas side-channel leakage only provides approximations (and since some of the squared Gram–Schmidt norms are rational numbers of very large height, recovering them exactly would require an unrealistic number of traces). As a result, the recovery algorithm has to be combined with some pruned tree search in order to account for approximate inputs. In practice, for the larger parameters of DLP signatures (with a claimed security level of 192 bits), we manage to recover the key with good probability using 2^{33} to 2^{35} DLP timing traces.

Carrying out such an experiment in the FALCON setting, however, is left as a challenging open problem for further work. This is because adapting the bit-reversed order totally positive recovery algorithm to deal with approximate inputs appears to be much more difficult (instead of sieving integers whose square lies in some specified interval, one would need to find the *cyclotomic* integers whose square lies in some target set, which does not even look simple to describe).

Related work. As noted above, the side-channel security of Fiat–Shamir lattice-based signature has been studied extensively, including in [10, 44, 16, 7, 5, 51]. However, the only implementation attacks we are aware of against hash-and-sign schemes are fault analysis papers [17, 36]: side-channel attacks have not been described so far to the best of our knowledge.

Aside from the original implementations of DLP and FALCON, which are the focus of this paper, several others have appeared in the literature. However, they usually do not aim for side-channel security [37, 42] or only make the *base* discrete Gaussian sampler (with fixed standard deviation) constant time [30], but do not eliminate the leakage of the varying standard deviations. As a result, those implementations are also vulnerable to the attacks of this paper.

This is not the case, however, for Pornin's very recent, updated implementation of FALCON, which uses a novel technique proposed by Prest, Ricosset and Rossi [49], combined with other recent results on constant time *rejection sampling* for discrete Gaussian distribution [54, 5] in order to eliminate the timing leakage of the lattice discrete Gaussian sampler. This technique applies to

discrete Gaussian sampling over $\mathbb{Z}$ with varying standard deviations, when those deviations only take values in a small range. It is then possible to eliminate the dependence on the standard deviation in the rejection sampling by scaling the target distribution to match the acceptance rate of the maximal possible standard deviation. The small range ensures that the overhead of this countermeasure is relatively modest. Thanks to this countermeasure, we stress that the most recent official implementation of FALCON is *already protected* against the attacks of this paper. Nevertheless, we believe our results underscore the importance of applying such countermeasures.

Organization of the paper. Following some preliminary material in Section 2, Section 3 is devoted to recalling some general facts about signature generation for hash-and-sign lattice-based schemes. Section 4 then gives a roadmap of our attack strategy, and provides some details about the final steps (how to deduce the secret key from the totally positive element $u = f\bar{f} + g\bar{g}$. Section 5 describes our main technical contribution: the algorithms that recover u from the Gram–Schmidt norms, both in the DLP and in the FALCON setting. Section 6 delves into the details of the side-channel leakage, showing how the implementations of the Gaussian samplers of DLP and FALCON do indeed reveal the Gram–Schmidt norms through timing side-channels. Finally, Section 7 presents our concrete experiments against DLP, including the tree search strategy to accomodate approximate Gram–Schmidt norms and experimental results in terms of timing and number of traces.

Notation. We use bold lowercase letters for vectors and bold uppercase for matrices. The zero vector is $\mathbf{0}$. We denote by $\mathbb{N}$ the non-negative integer set and by $\log$ the natural logarithm. Vectors are in row form, and we write $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ to denote that $\mathbf{b}_i$ is the i-th row of $\mathbf{B}$. For a matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$, we denote by $\mathbf{B}_{i,j}$ the entry in the i-th row and j-th column of $\mathbf{B}$, where $i \in \{0, \ldots, n-1\}$ and $j \in \{0, \ldots, m-1\}$. For $I \subseteq [0, n), J \subseteq [0, m)$, we denote by $\mathbf{B}_{I \times J}$ the submatrix $(\mathbf{B}_{i,j})_{i \in I, j \in J}$. In particular, we write $\mathbf{B}_I = \mathbf{B}_{I \times I}$. Let $\mathbf{B}^t$ denote the transpose of $\mathbf{B}$.

Given $\mathbf{u} = (u_0, \ldots, u_{n-1})$ and $\mathbf{v} = (v_0, \ldots, v_{n-1})$, their inner product is $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=0}^{n-1} u_i v_i$. The ℓ_2-norm of $\mathbf{v}$ is $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$ and the ℓ_∞-norm is $\|\mathbf{v}\|_\infty = \max_i |v_i|$. Let $\det(\mathbf{B})$ be the determinant of $\mathbf{B}$. For $\mathbf{B} \in \mathbb{R}^{n \times n}$, we call $\det\left(\mathbf{B}_{[0,i]}\right)$ the i-th leading principal minor of $\mathbf{B}$ where $0 \leq i \leq n-1$.

Let D be a distribution. We write $z \hookleftarrow D$ when the random variable z is sampled from D, and denote by $D(x)$ the probability of $z = x$. For a random variable z, its expectation is $\mathbb{E}[z]$. We write $\mathcal{N}(\mu, \sigma^2)$ the normal distribution of mean μ and variance σ^2. For any finite set S, we let $U(S)$ denote the uniform distribution over S. For a real-valued function f and any countable set S in the domain of f, we write $f(S) = \sum_{x \in S} f(x)$.

2 Preliminaries

A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. If it is generated by $\mathbf{B} \in \mathbb{R}^{n \times m}$, we also write $\mathcal{L} := \mathcal{L}(\mathbf{B}) = \{\mathbf{x}\mathbf{B} \mid \mathbf{x} \in \mathbb{Z}^n\}$. If $\mathbf{B}$ has full row rank, then we call $\mathbf{B}$ a basis and n the rank of $\mathcal{L}$.

2.1 Gram–Schmidt Orthogonalization

Let $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}) \in \mathbb{Q}^{n \times m}$ of rank n. The Gram-Schmidt orthogonalization of $\mathbf{B}$ is $\mathbf{B} = \mathbf{L}\mathbf{B}^*$, where $\mathbf{L} \in \mathbb{Q}^{n \times n}$ is lower-triangular with 1 on its diagonal and $\mathbf{B}^* = (\mathbf{b}_0^*, \ldots, \mathbf{b}_{n-1}^*)$ is

a matrix with pairwise orthogonal rows. We call $\|\mathbf{b}_i^*\|$ the i-th Gram-Schmidt norm of $\mathbf{B}$, and let $\|\mathbf{B}\|_{GS} = \max_i \|\mathbf{b}_i^*\|$.

The Gram matrix of $\mathbf{B}$ is $\mathbf{G} = \mathbf{B}\mathbf{B}^t$, and satisfies $\mathbf{G} = \mathbf{L}\mathbf{D}\mathbf{L}^t$ where $\mathbf{D} = \operatorname{diag}\left(\|\mathbf{b}_i^*\|^2\right)$. This is also known as the Cholesky decomposition of $\mathbf{G}$, and such a decomposition exists for any symmetric positive definite matrix. The next proposition follows from the triangular structure of $\mathbf{L}$.

Proposition 1. *Let $\mathbf{B} = \mathbb{Q}^{n\times m}$ of rank n and $\mathbf{G}$ its Gram matrix. Then for all integer $0 \leq k \leq n-1$, we have $\det\left(\mathbf{G}_{[0,k]}\right) = \prod_{i=0}^{k} \|\mathbf{b}_i^*\|^2$.*

Let $\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}$, where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{D} \in \mathbb{R}^{m\times m}$ are invertible matrices, then $\mathbf{M}/\mathbf{A} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \in \mathbb{R}^{m\times m}$ is called the Schur complement. It holds that

$$\det(\mathbf{M}) = \det(\mathbf{A}) \det(\mathbf{M}/\mathbf{A}). \tag{1}$$

2.2 Parametric Statistics

Let D_p be some distribution determined by parameter p. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a vector of observed samples of $X \hookleftarrow D_p$. The *log-likelihood function* with respect to $\mathbf{X}$ is

$$\ell_{\mathbf{X}}(p) = \sum_{i=1}^{n} \log(D_p(X = X_i)).$$

Provided the log-likelihood function is bounded, a *maximum likelihood estimator* for samples $\mathbf{X}$ is a real $\mathrm{MLE}(\mathbf{X})$ maximizing $\ell_{\mathbf{X}}(p)$. The *Fisher information* is

$$\mathcal{I}(p) = -\,\mathbb{E}\left[\frac{d^2}{dp^2}\ell_X(p)\right].$$

Seen as a random variable, it is known (e.g. [27, Theorem 6.4.2]) that $\sqrt{n}(\mathrm{MLE}(\mathbf{X}) - p)$ converges in distribution to $\mathcal{N}(0, \mathcal{I}(p)^{-1})$. When the target distribution is a geometric, maximum likelihood estimators and the Fisher information are well-known. The second statement of the next lemma directly comes from a Gaussian tail bound.

Lemma 1. *Let Geo_p denote a geometric distribution with parameter p, and $\mathbf{X} = (X_1, \cdots, X_n)$ be samples from Geo_p. Then we have $MLE(\mathbf{X}) = \frac{n}{\sum_{i=1}^{n} X_i}$ and $\sqrt{n}(MLE(\mathbf{X}) - p)$ converges in distribution to $\mathcal{N}(0, p^2(1-p))$. In particular, when N is large, then for any $\alpha \geq 1$, we have $|MLE(\mathbf{X}) - p| \leq \alpha \cdot p\sqrt{\frac{1-p}{N}}$ except with probability at most $2\exp(-\alpha^2/2)$.*

2.3 Discrete Gaussian Distributions

Let $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}\right)$ be the n-dimensional Gaussian function with center $\mathbf{c} \in \mathbb{R}^n$ and standard deviation σ. When $\mathbf{c} = \mathbf{0}$, we just write $\rho_\sigma(\mathbf{x})$. The discrete Gaussian over a lattice $\mathcal{L}$ with center $\mathbf{c}$ and standard deviation parameter σ is defined by the probability function

$$D_{\mathcal{L},\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\mathcal{L})}, \forall \mathbf{x} \in \mathcal{L}.$$

In this work, the case $\mathcal{L} = \mathbb{Z}$ is of particular interest. It is well known that $\int_{-\infty}^{\infty} \rho_{\sigma,c}(x)\mathrm{d}x = \sigma\sqrt{2\pi}$. Notice that $D_{\mathbb{Z},\sigma,c}$ is equivalent to $i + D_{\mathbb{Z},\sigma,c-i}$ for an arbitrary $i \in \mathbb{Z}$, hence it suffices to consider the case where $c \in [0, 1)$. The half discrete integer Gaussian, denoted by $D^{+}_{\mathbb{Z},\sigma,c}$, is defined by

$$D^{+}_{\mathbb{Z},\sigma,c}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(\mathbb{N})}, \forall x \in \mathbb{N}.$$

We again omit the center when it is $c = 0$. For any $\epsilon > 0$, the (scaled)[4] smoothing parameter $\eta'_{\epsilon}(\mathbb{Z})$ is the smallest $s > 0$ such that $\rho_{1/s\sqrt{2\pi}}(\mathbb{Z}) \leq 1 + \epsilon$. In practice, ϵ is very small, say 2^{-50}. The smoothing parameter allows to quantify precisely how the discrete Gaussian differs from the standard Gaussian function.

Lemma 2 ([39], implicit in Lemma 4.4). *If $\sigma \geq \eta'_{\epsilon}(\mathbb{Z})$, then $\rho_{\sigma}(c + \mathbb{Z}) \in [\frac{1-\epsilon}{1+\epsilon}, 1]\rho_{\sigma}(\mathbb{Z})$ for any $c \in [0, 1)$.*

Corollary 1. *If $\sigma \geq \eta'_{\epsilon}(\mathbb{Z})$, then $\rho_{\sigma}(\mathbb{Z}) \in [1, \frac{1+\epsilon}{1-\epsilon}]\sqrt{2\pi}\sigma$.*

Proof. Notice that $\int_0^1 \rho_{\sigma}(\mathbb{Z}+c)\mathrm{d}c = \int_{-\infty}^{\infty} \rho_{\sigma}(x)\mathrm{d}x = \sqrt{2\pi}\sigma$, the proof is completed by Lemma 2. □

2.4 Power-of-Two Cyclotomic Fields

For the rest of this article, we let $n = 2^{\ell}$ for some integer $\ell \geq 1$. We let ζ_n be a $2n$-th primitive root of 1. Then $\mathcal{K}_n = \mathbb{Q}(\zeta_n)$ is the n-th power-of-two cyclotomic field, and comes together with its ring of algebraic integers $\mathcal{R}_n = \mathbb{Z}[\zeta_n]$. It is also equipped with n field automorphisms forming the Galois group which is commutative in this case. It can be seen that $\mathcal{K}_{n/2} = \mathbb{Q}(\zeta_{n/2})$ is the subfield of $\mathcal{K}_n$ fixed by the automorphism $\sigma(\zeta_n) = -\zeta_n$ of $\mathcal{K}_n$, as $\zeta_n^2 = \zeta_{n/2}$. This leads to a tower of field extensions and their corresponding ring of integers

$$\begin{array}{ccccccc} \mathcal{K}_n & \supseteq & \mathcal{K}_{n/2} & \supseteq \cdots \supseteq & \mathcal{K}_1 & = & \mathbb{Q} \\ \cup & & \cup & \cdots & \cup & & \\ \mathcal{R}_n & \supseteq & \mathcal{R}_{n/2} & \supseteq \cdots \supseteq & \mathcal{R}_1 & = & \mathbb{Z} \end{array}$$

Given an extension $\mathcal{K}_n|\mathcal{K}_{n/2}$, the relative trace $\mathrm{Tr} : \mathcal{K}_n \to \mathcal{K}_{n/2}$ is the $\mathcal{K}_{n/2}$-linear map given by $\mathrm{Tr}(f) = f + \sigma(f)$. Similarly, the relative norm is the multiplicative map $\mathrm{N}(f) = f \cdot \sigma(f) \in \mathcal{K}_{n/2}$. Both maps send integers in $\mathcal{K}_n$ to integers in $\mathcal{K}_{n/2}$. For all $f \in \mathcal{K}_n$, it holds that $f = (1/2)(\mathrm{Tr}(f) + \zeta_n \mathrm{Tr}(\zeta_n^{-1} f))$.

We are also interested in the field automorphism $\zeta_n \mapsto \zeta_n^{-1} = \bar{\zeta_n}$, which corresponds to the complex conjugation. We call *adjoint* the image $\bar{f}$ of f under this automorphism. The fixed subfield $\mathcal{K}_n^{+} := \mathbb{Q}(\zeta_n + \zeta_n^{-1})$ is known as the *totally real subfield* and contains the *self-adjoint* elements, that is, such that $f = \bar{f}$. Another way to describe self-adjoint elements is to say that all their complex embeddings[5] are in fact reals. Elements whose embeddings are all positive are called *totally positive elements*, and we denote their set by $\mathcal{K}_n^{++}$. A standard example of such an element is given by $f\bar{f}$ for any $f \in \mathcal{K}_n$. It is well-known that the Galois automorphisms act as permutation of these embeddings, so that a totally positive element stays positive under the action of the Galois group.

[4] The scaling factor is $(\sqrt{2\pi})^{-1}$ before the smoothing parameter $\eta_{\epsilon}(\mathbb{Z})$ in [39].

[5] Each root of $x^n + 1$ describes one complex embedding by mean of evaluation.

Representation of cyclotomic numbers We also have $\mathcal{K}_n \simeq \mathbb{Q}[x]/(x^n+1)$ and $\mathcal{R}_n \simeq \mathbb{Z}[x]/(x^n+1)$, so that elements in cyclotomic fields can be seen as polynomials. In this work, each $f = \sum_{i=0}^{n-1} f_i \zeta_n^i \in \mathcal{K}_n$ is identified with its coefficient vector $(f_0, \cdots, f_{n-1})$. Then the inner product of f and g is $\langle f, g\rangle = \sum_{i=0}^{n-1} f_i g_i$, and we write $\|f\|$, resp. $\|f\|_\infty$, the ℓ_2-norm, resp. ℓ_∞-norm, of f. In this representation, it can be checked that $\bar{f} = (f_0, -f_{n-1}, \ldots, -f_1)$ and that $\langle f, gh\rangle = \langle f\bar{g}, h\rangle$ for all $f, g, h \in \mathcal{K}_n$. In particular, the constant coefficient of fg is $\langle f, g\rangle = \langle f\bar{g}, 1\rangle$. A self-adjoint element f has coefficients $(f_0, f_1, \ldots, f_{n/2-1}, 0, -f_{n/2-1}, \ldots, -f_1)$.

Elements in $\mathcal{K}_n$ can also be represented by their matrix of multiplication in the basis $1, \zeta_n, \ldots, \zeta_n^{n-1}$. In other words, the map $\mathcal{A}_n : \mathcal{K}_n \to \mathbb{Q}^{n\times n}$ defined by

$$\mathcal{A}_n(f) = \begin{pmatrix} f_0 & f_1 & \cdots & f_{n-1} \\ -f_{n-1} & f_0 & \cdots & f_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ -f_1 & -f_2 & \cdots & f_0 \end{pmatrix} = \begin{pmatrix} f \\ \zeta_n \cdot f \\ \vdots \\ \zeta_n^{n-1} \cdot f \end{pmatrix}$$

is a ring isomorphism. We have $fg = g \cdot \mathcal{A}_n(f)$. We can also see that $\mathcal{A}_n(\bar{f}) = \mathcal{A}_n(f)^t$ which justifies the term "adjoint". We deduce that the matrix of a self-adjoint element is symmetric. It can be observed that a totally positive element $A \in \mathcal{K}_n$ corresponds to the symmetric positive definite matrix $\mathcal{A}_n(A)$.

For efficiency reasons, the scheme FALCON uses another representation corresponding to the tower structure. If $f = (f_0, \ldots, f_{n-1}) \in \mathcal{K}_n$, we let $f_e = (1/2)\operatorname{Tr}(f) = (f_0, f_2, \ldots, f_{n-2})$ and $f_o = (1/2)\operatorname{Tr}(\zeta_n^{-1} f) = (f_1, f_3, \ldots, f_{n-1})$. Let $\mathbf{P}_n \in \mathbb{Z}^{n\times n}$ be the permutation matrix corresponding to the bit-reversal order. We define $\mathcal{F}_n(f) = \mathbf{P}_n \mathcal{A}_n(f) \mathbf{P}_n^t$. In particular, it is also symmetric positive definite when f is a totally positive element. As shown in [15], it holds that

$$\mathcal{F}_n(f) = \begin{pmatrix} \mathcal{F}_{n/2}(f_e) & \mathcal{F}_{n/2}(f_o) \\ \mathcal{F}_{n/2}(\zeta_{n/2} f_o) & \mathcal{F}_{n/2}(f_e) \end{pmatrix}. \tag{2}$$

2.5 NTRU Lattices

Given $f, g \in \mathcal{R}_n$ such that f is invertible modulo some $q \in \mathbb{Z}$, we let $h = f^{-1} g \bmod q$. The NTRU lattice determined by h is $\mathcal{L}_{\text{NTRU}} = \{(u, v) \in \mathcal{R}_n^2 \,:\, u + vh = 0 \bmod q\}$. Two bases of this lattice are of particular interest for cryptography:

$$\mathbf{B}_{\text{NTRU}} = \begin{pmatrix} q & 0 \\ -h & 1 \end{pmatrix} \text{ and } \mathbf{B}_{f,g} = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix},$$

where $F, G \in \mathcal{R}_n$ such that $fG - gF = q$. Indeed, the former basis acts usually as the public key, while the latter is the secret key, also called the trapdoor basis, when f, g, F, G are short vectors. In practice, these matrices are represented using either the operator $\mathcal{A}_n$ [13] or $\mathcal{F}_n$ [48]:

$$\mathbf{B}_{f,g}^{\mathcal{A}} = \begin{pmatrix} \mathcal{A}_n(g) & \mathcal{A}_n(-f) \\ \mathcal{A}_n(G) & \mathcal{A}_n(-F) \end{pmatrix} \quad \text{and} \quad \mathbf{B}_{f,g}^{\mathcal{F}} = \begin{pmatrix} \mathcal{F}_n(g) & \mathcal{F}_n(-f) \\ \mathcal{F}_n(G) & \mathcal{F}_n(-F) \end{pmatrix}.$$

3 Hash-and-Sign over NTRU Lattices

Gentry, Peikert and Vaikuntanathan introduced in [21] a generic and provably secure hash-and-sign framework based on trapdoor sampling. This paradigm has then been instantiated over NTRU

lattices giving rise to practically efficient cryptosystems: DLP [13] and FALCON [48] signature schemes.

In the NTRU-based hash-and-sign scheme, the secret key is a pair of short polynomials $(f, g) \in \mathcal{R}_n^2$ and the public key is $h = f^{-1}g \bmod q$. The *trapdoor basis* $\mathbf{B}_{f,g}$ (of $\mathcal{L}_{\text{NTRU}}$) derives from (f, g) by computing $F, G \in \mathcal{R}_n$ such that $fG - gF = q$. In both the DLP signature and FALCON, the trapdoor basis has a bounded Gram-Schmidt norm: $\|\mathbf{B}_{f,g}\|_{GS} \leq 1.17\sqrt{q}$ for compact signatures.

The signing and verification procedure is described on a high level as follows:

Sign$(m, \mathsf{sk} = \mathbf{B}_{f,g})$
Compute $c = \mathsf{hash}(m) \in \mathcal{R}_n$;
Using sk, sample a short (s_1, s_2)
such that $s_1 + s_2 h = c \bmod q$;
Return $s = s_2$.

Verify$(m, s, \mathsf{pk} = h)$
Compute $c = \mathsf{hash}(m) \in \mathcal{R}_n$;
Compute $\mathbf{s} = (c - sh \bmod q, s)$;
If $\|\mathbf{s}\|$ is not small enough, reject.
Accept.

Lattice Gaussian samplers [21, 43] are nowadays a standard tool to generate signatures provably statistically independent of the secret basis. However, such samplers are also a notorious target for side-channel attacks. This work makes no exception and attacks non constant-time implementations of the lattice Gaussian samplers at the heart of both DLP and FALCON, that are based on the KGPV sampler [31] or its ring variant [15]. Precisely, while previous attacks target to Gaussian with *public* standard deviations, our attack learns the *secret-dependent* Gaussian standard deviations involved in the KGPV sampler.

3.1 The KGPV sampler and its variant

The KGPV sampler is a randomized variant of Babai's nearest plane algorithm [1]: instead of rounding each center to the closest integer, the KGPV sampler determines the integral coefficients according to some integer Gaussians. It is shown in [21] that under certain smoothness condition, the algorithm outputs a sample from a distribution negligibly close to the target Gaussian. Its formal description is illustrated in Algorithm 3.1.

Note that in the KGPV sampler (or its ring variant), the standard deviations of integer Gaussians are inversely proportional to the Gram-Schmidt norms of the input basis. In the DLP scheme, $\mathbf{B}$ is in fact the trapdoor basis $\mathbf{B}_{f,g}^{\mathcal{A}} \in \mathbb{Z}^{2n \times 2n}$.

The Ducas–Prest sampler. FALCON uses a variant of the KGPV algorithm which stems naturally from Ducas–Prest's fast Fourier nearest plane algorithm [15]. It exploits the tower structure of power-of-two cyclotomic rings. Just like the KGPV sampler, the Ducas-Prest sampler fundamentally relies on integer Gaussian sampling to output Gaussian vectors. We omit its algorithmic description, as it is not needed in this work. Overall, what matters is to understand that the standard deviations of involved integer Gaussians are also in the form $\sigma_i = \sigma / \|\mathbf{b}_i^*\|$, but that $\mathbf{B} = \mathbf{B}_{f,g}^{\mathcal{F}}$ in this context.

4 Side-Channel Attack against Trapdoor Samplers: a roadmap

Our algorithm proceeds as follows:

1. Side-channel leakage: extract the $\|\mathbf{b}_i^*\|$'s associated to $\mathbf{B}_{f,g}^{\mathcal{A}}$, resp. $\mathbf{B}_{f,g}^{\mathcal{F}}$ via the timing leakage of integer Gaussian sampler in the DLP scheme, reps. FALCON.

Algorithm 3.1 The KGPV algorithm $\mathsf{KGPV}(\sigma, \mathbf{B}, \mathbf{c})$

Input: a basis $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{n-1})$ of a lattice $\mathcal{L}$, $\mathbf{c} \in \mathbb{Q}^n$ and $\sigma \geq \|\mathbf{B}\|_{GS} \cdot \eta_\epsilon(\mathbb{Z})$.
Output: $\mathbf{z} \in \mathbb{Z}^n$ such that $\mathbf{zB}$ follows a distribution close to $D_{\mathcal{L},\sigma,\mathbf{cB}^*}$.

Precomputation:
compute $\mathbf{B} = \mathbf{LB}^*$
$(\boldsymbol{\mu_0}, \cdots, \boldsymbol{\mu_{n-1}}) \leftarrow \mathbf{L} - \mathbf{I}_n$
for $i = n-1, \cdots, 0$ **do**
  $\sigma_i \leftarrow \sigma / \|\mathbf{b}_i^*\|$
end for
Sampling:
$\mathbf{z} \leftarrow \mathbf{0}, \mathbf{c}' \leftarrow \mathbf{c}$
for $i = n-1, \cdots, 0$ **do**
  $z_i \leftarrow \mathsf{GaussianIntegerSampler}(\sigma_i, c_i')$
  $\mathbf{c}' \leftarrow \mathbf{c}' - z_i \boldsymbol{\mu_i}$
end for
return $\mathbf{z}$

2. Totally positive recovery: from the given $\|\mathbf{b}_i^*\|$'s, recover a conjugate u of $f\overline{f} + g\overline{g} \in \mathcal{K}_n^{++}$.
3. Final recovery: compute f from u and the public key $g/f \mod q$.

Steps 1 and 2 of the attack are the focus of sections 6 and 5 respectively. Below we describe how the third step is performed. First we recover the element $f\overline{g}$, using the fact that it has small coefficients. More precisely, the j^{th} coefficient is $\langle f, \zeta_n^j g \rangle$ where f and $\zeta_n^j g$ are independent and identically distributed according to $D_{\mathbb{Z}^n,r}$, with $r = 1.17\sqrt{\frac{q}{2n}}$. By [33, Lemma 4.3], we know that all these coefficients are of size much smaller than $q/2$ with high probability. Now, we can compute $v = u\overline{h}(1 + h\overline{h})^{-1} \bmod q$, where $h = f^{-1}g \bmod q$ is the public verification key. We readily see that $v = f\overline{g} \bmod q$ if and only if $u = f\overline{f} + g\overline{g}$. If u is a conjugate of $f\overline{f} + g\overline{g}$, then most likely the coefficients of v will look random in $(-q/2, q/2]$. This can mostly be interpreted as the NTRU assumption, that is, h being indistinguishable from a random element modulo q. When this happens, we just consider another conjugate of u, until we obtain a distinguishably small element, which must then be $f\overline{g}$ (not just in reduction modulo q, but in fact over the integers).

Once this is done, we can then deduce the reduction modulo q of $f\bar{f} \equiv f\bar{g}/\bar{h} \pmod q$, which again coincides with $f\bar{f}$ over the integers with high probability (if we again lift elements of $\mathbb{Z}_q$ to $(-q/2, q/2]$, except for the constant coefficient, which should be lifted positively). This boils down to the fact that with high probability $f\overline{f}$ has its constant coefficient in $(0, q)$ and the others are in $(-q/2, q/2)$. Indeed, the constant coefficient of $f\overline{f}$ is $\|f\|^2$, and the others are $\langle f, \zeta_n^j f \rangle$'s with $j \geq 1$. By some Gaussian tail bound, we can show $\|f\|^2 \leq q$ with high probability. As for $\langle f, \zeta_n^j f \rangle$'s, despite the dependency between f and $\zeta_n^j f$, we can still expect $|\langle f, \zeta_n^j f \rangle| < q/2$ for all $j \geq 1$ with high probability. We leave details in Supplementary Material B for interested readers.

Next, we compute the ideal (f) from the knowledge of $f\overline{f}$ and $f\overline{g}$. Indeed, as f and g are co-prime from the key generation algorithm, we directly have $(f) = (f\overline{f}) + (f\overline{g})$. At this point, we have obtained both the ideal (f) and the relative norm $f\bar{f}$ of f on the totally real subfield. That data is exactly what we need to apply the Gentry–Szydlo algorithm [22], and finally recover f itself in polynomial time. Note furthermore that the practicality of the Gentry–Szydlo algorithm for the dimensions we consider ($n = 512$) has been validated in previous work [16].

Comparison with existing method. As part of their side-channel analysis of the BLISS signature scheme, Espitau *et al.* [16] used the Howgrave-Graham–Szydlo algorithm to recover an NTRU secret f from $f\overline{f}$. They successfully solved a small proportion ($\approx 7\%$) of NTRU instances with

$n = 512$ in practice. The Howgrave-Graham–Szydlo algorithm first recovers the ideal (f) and then calls the Gentry–Szydlo algorithm as we do above. The bottleneck of this method is in its reliance on integer factorization for ideal recovery: the integers involved can become quite large for an arbitrary f, so that recovery cannot be done in classical polynomial time in general. This is why only a small proportion of instances can be solved in practice.

However, the technique we describe above bypasses this expensive factorization step by exploiting the arithmetic property of the NTRU secret key. In particular, it is immediate to obtain a two-element description of (f), so that the Gentry-Szydlo algorithm can be run as soon as $f\bar{f}$ and $f\bar{g}$ are computed. This significantly improves the applicability and efficiency of Espitau *et al.*'s side-channel attack against BLISS [16]. The question of avoiding the reliance on Gentry–Szydlo algorithm by using the knowledge of $f\bar{g}$ and $f\bar{f}$ remains open, however.

5 Recovering Totally Positive Elements

Totally positive elements in $\mathcal{K}_n$ correspond to symmetric positive definite matrices with an inner structure coming from the algebra of the field. In particular, it is enough to know only one line of the matrix to recover the corresponding field element. Hence it can be expected that being given the diagonal part of the LDL decomposition also suffices to perform a recovery. In this section, we show that this is indeed the case provided we know *exactly* the diagonal.

Recall on the one hand that the $\mathcal{A}_n$ representation is the skew circulant matrix in which each diagonal consists of the same entries. On the other hand, the $\mathcal{F}_n$ representation does not follow the circulant structure, but it is compatible with the tower of rings structure, i.e. its sub-matrices are the $\mathcal{F}_{n/2}$ representations of elements in the subfield $\mathcal{K}_{n/2}$. Each operator leads to a distinct approach, which is described in section 5.1 and 5.2 respectively.

While the algorithms of this section can be used independently, they are naturally related to hash-and-sign over NTRU lattices. Let $\mathbf{B}$ be a matrix representation of some secret key $(g, -f)$, and $\mathbf{G} = \mathbf{B}\mathbf{B}^t$. Then the diagonal part of $\mathbf{G}$'s LDL decomposition contains the $\|\mathbf{b}_i^*\|$'s, and $\mathbf{G}$ is a matrix representation of $f\bar{f} + g\bar{g} \in \mathcal{K}_n^{++}$. As illustrated in Section 4, the knowledge of $u = f\bar{f} + g\bar{g}$ allows to recover the secret key in polynomial time. Therefore results in this section pave the way for a better use of secret Gram-Schmidt norms.

In practice however, we will obtain only *approximations* of the $\|\mathbf{b}_i^*\|$'s. The algorithms of this section must then be tweaked to handle the approximation error. The case of $\mathcal{A}_n$ is dealt with in Section 7.1. While we do not solve the "approximate" case of $\mathcal{F}_n$, we believe our "exact" algorithms to be of independent interest to the community.

5.1 Case of the Power Basis

The goal of this section is to obtain the next theorem. It involves the heuristic argument that some rational quadratic equations always admits exactly one integer root, which will correspond to a coefficient of the recovered totally positive element. Experimentally, when it happens that there are two integer roots and the wrong one is chosen, the algorithm "fails" with overwhelming probability at the next step: the next discriminant does not lead to integer roots.

Theorem 1. *Let $u \in \mathcal{R}_n \cap \mathcal{K}_n^{++}$. Write $\mathcal{A}_n(u) = \mathbf{L} \cdot \mathrm{diag}(\lambda_i)_i \cdot \mathbf{L}^t$. There is a (heuristic) algorithm* $\mathsf{Recovery}_{\mathcal{A}}$ *that, given λ_i's, computes u or $\sigma(u)$. It runs in $\widetilde{O}(n^3 \log \|u\|_\infty)$.*

Algorithm 5.1 $\mathsf{Recovery}_{\mathcal{A}}(m_0, \ldots, m_{n-1})$

Input: $m_0, \ldots, m_{n-1} \in \mathbb{Z}_+$.
Output: $u \in \mathcal{R}_n$ such that u is totally positive and the principal minors of $\mathcal{F}_n(u)$ are m_i's ($0 \leq i < n$).

$u_0 \leftarrow m_0$
$u_1 \leftarrow$ any root of $u_0 - \frac{m_1}{m_0} - \frac{X^2}{u_0}$
for $i = 1$ to $n/2 - 2$ **do**
  Build $\mathcal{A}_n(u)_{[0,i]}$ from $u_i, \ldots, u_0$
  $\mathbf{v}_i \leftarrow (X, u_i, \ldots, u_1)$
  Solve $\mathcal{A}_n(u)_{[0,i]} \cdot \mathbf{w}_i^t = \mathbf{v}_i^t$ for $\mathbf{w}_i$
  $E \leftarrow u_0 - m_{i+1}/m_i - \mathbf{v}_i \cdot \mathbf{w}_i^t$.
  Compute the roots $\{r_1, r_2\}$ of E
  $u_{i+1} \leftarrow \{r_1, r_2\} \cap \mathbb{Z}$
end for
return $(u_0, u_1, \ldots, u_{n/2-1}, 0, -u_{n/2-1}, \ldots, -u_1)$.

The complexity analysis is given in Supplementary Material A.1. In Section 7.2, a version tweaked to handle approximations of the λ_i's is given, and may achieve quasi-quadratic complexity. It is in any case very efficient in practice, and it is used in our attack against DLP signature.

We now describe Algorithm 5.1. It holds that $u_0 = \det\left(\mathcal{A}_n(u)_{[0,0]}\right) = \lambda_0$. By the self-adjointness of u, we only need to consider the first $n/2$ coefficients. For any $0 \leq i < n/2 - 1$, we have

$$\mathcal{A}_n(u)_{[0,i+1]} = \begin{pmatrix} & & & u_{i+1} \\ & \mathcal{A}_n(u)_{[0,i]} & & \vdots \\ & & & u_1 \\ u_{i+1} & \ldots & u_1 & u_0 \end{pmatrix}.$$

Let $\mathbf{v}_i = (u_{i+1}, \ldots, u_1)$. By the definition of the Schur complement and Proposition 1, we see that

$$\frac{\det\left(\mathcal{A}_n(u)_{[0,i+1]}\right)}{\det\left(\mathcal{A}_n(u)_{[0,i]}\right)} = u_0 - \mathbf{v}_i \mathcal{A}_n(u)_{[0,i]}^{-1} \mathbf{v}_i^t,$$

where the left-hand side is actually λ_{i+1}, and the right-hand side gives a quadratic equation in u_{i+1} with rational coefficients that can be computed from the knowledge of $(u_0, \ldots, u_i)$. When $i = 0$, the equation is equivalent to $\lambda_0 \lambda_1 = u_0^2 - u_1^2$: there are two candidates of u_1 up to sign. Once u_1 is chosen, for $i \geq 1$, the quadratic equation should have with very high probability a unique integer solution, i.e. the corresponding u_{i+1}. This leads to Algorithm 5.1. Note that the sign of u_1 determines whether the algorithm recovers u or $\sigma(u)$. This comes from the fact that $\mathcal{A}_n(u) = \mathrm{diag}((-1)^i)_{i \leq n} \cdot \mathcal{A}_n(\sigma(u)) \cdot \mathrm{diag}((-1)^i)_{i \leq n}$.

5.2 Case of the Bit-Reversed Order Basis

In this section, we are given the diagonal part of the LDL decomposition $\mathcal{F}_n(u) = \mathbf{L}' \,\mathrm{diag}(\lambda_i) \mathbf{L}'^t$, which rewrites as $(\mathbf{L}'^{-1}\mathbf{P}_n)\mathcal{A}_n(u)(\mathbf{L}'^{-1}\mathbf{P}_n)^t = \mathrm{diag}(\lambda_i)$. Since the triangular structure is shuffled by the bit-reversal representation, recovering u from the λ_i's is not as straightforward as in the previous section. Nevertheless, the compatibility of the $\mathcal{F}_n$ operator with the tower of extension can be exploited. It gives a recursive approach that stems from natural identities between the trace and norm maps relative to the extension $\mathcal{K}_n \,|\, \mathcal{K}_{n/2}$, crucially uses the self-adjointness and total positivity of u, and fundamentally relies on computing square roots in $\mathcal{R}_n$.

Theorem 2. *Let $u \in \mathcal{R}_n \cap \mathcal{K}_n^{++}$. Write $\mathcal{F}_n(u) = \mathbf{L}' \cdot \mathrm{diag}(\lambda_i)_i \cdot \mathbf{L}'^t$. There is a (heuristic) algorithm that, given the λ_i's, computes a conjugate of u. It runs in $\widetilde{O}(n^3 \log \|u\|_\infty)$.*

The recursiveness of the algorithm and its reliance on square roots will force it to always work "up to Galois conjugation". In particular, at some point we will assume heuristically that only one of the conjugates of a value computed within the algorithm is in a given coset of the subgroup of relative norms in the quadratic subfield. Since that constraint only holds with negligible probability for random values, the heuristic is essentially always verified in practice. Recall that we showed in Section 4 how to recover the needed conjugate in practice by a distinguishing argument.

The rest of the section describes the algorithm, while the complexity analysis is presented in Supplementary Material A.2. First, we observe from

$$\mathrm{Tr}(u) + \zeta_n \,\mathrm{Tr}(\zeta_n^{-1} u) = 2u = 2\bar{u} = \overline{\mathrm{Tr}(u)} + \zeta_n^{-1}\overline{\mathrm{Tr}(\zeta_n^{-1} u)}$$

that $\mathrm{Tr}(u)$ is self-adjoint. The positivity of u implies that $\mathrm{Tr}(u) \in \mathcal{K}_{n/2}^{++}$. From Equation (2), we know that the $n/2$ first minors of $\mathcal{F}_n(u)$ are the minors of $\mathcal{F}_{n/2}(\mathrm{Tr}(u)/2)$. The identity above also shows that $\mathrm{Tr}(\zeta_n^{-1}u)$ is a square root of the element $\zeta_{n/2}^{-1}\,\mathrm{Tr}(\zeta_n^{-1}u)\overline{\mathrm{Tr}(\zeta_n^{-1}u)}$ in $\mathcal{K}_{n/2}$. Thus, if we knew $\mathrm{Tr}(\zeta_n^{-1}u)\overline{\mathrm{Tr}(\zeta_n^{-1}u)}$, we could reduce the problem of computing $u \in \mathcal{K}_n$ to computations in $\mathcal{K}_{n/2}$, more precisely, recovering a totally positive element from "its minors" and a square root computation.

It turns out that $\mathrm{Tr}(\zeta_n^{-1}u)\overline{\mathrm{Tr}(\zeta_n^{-1}u)}$ can be computed by going down the tower as well. One can see that

$$\mathrm{Tr}(u)^2 - 4\,\mathrm{N}(u) = \mathrm{Tr}(\zeta_n^{-1}u)\overline{\mathrm{Tr}(\zeta_n^{-1}u)}, \tag{3}$$

where $\mathrm{N}(u)$ is totally positive since u (and therefore $\sigma(u)$) is. This identity[6] can be thought as a "number field version" of the $\mathcal{F}_n$ representation. Indeed, recall that $u_e = (1/2)\,\mathrm{Tr}(u)$ and $u_o = (1/2)\,\mathrm{Tr}(\zeta_n^{-1}u)$. Then by block determinant formula and the fact that $\mathcal{F}_n$ is a ring isomorphism, we see that

$$\det \mathcal{F}_n(u) = \prod_{i=0}^{n-1} \lambda_i = \det(\mathcal{F}_{n/2}(u_e)^2 - \mathcal{F}_{n/2}(u_o\overline{u_o})).$$

This strongly suggests a link between the successive minors of $\mathcal{F}_n(u)$ and the element $\mathrm{N}(u)$. The next lemma makes this relation precise, and essentially amounts to taking Schur complements in the above formula.

Lemma 3. *Let $u \in \mathcal{K}_n^{++}$ and $\widehat{u} = \frac{2\,\mathrm{N}(u)}{\mathrm{Tr}(u)} \in \mathcal{K}_{n/2}^{++}$. Then for $0 < k < n/2$, we have*

$$\det\left(\mathcal{F}_n(u)_{[0,k+\frac{n}{2})}\right) = \det\left(\mathcal{F}_{n/2}(u_e)\right)\det\left(\mathcal{F}_{n/2}(\widehat{u})_{[0,k)}\right).$$

Proof. Let $\mathbf{G} = \mathcal{F}_n(u)$ and $\mathbf{B} = \mathcal{F}_{n/2}(u_o)_{[0,\frac{n}{2})\times[0,k)}$ in order to write

$$\mathbf{G}_{[0,\frac{n}{2}+k)} = \begin{pmatrix} \mathcal{F}_{n/2}(u_e) & \mathbf{B} \\ \mathbf{B}^t & \mathcal{F}_{n/2}(u_e)_{[0,k)} \end{pmatrix},$$

with $\mathbf{B}^t = \mathcal{F}_{n/2}(\overline{u_o})_{[0,k)\times[0,\frac{n}{2})}$. Let $\mathbf{S} = \mathbf{G}_{[0,\frac{n}{2}+k)}/\mathcal{F}_{n/2}(u_e) = \mathcal{F}_{n/2}(u_e)_{[0,k)} - \mathbf{B}\mathcal{F}_{n/2}(u_e)^{-1}\mathbf{B}^t$. Since $\mathcal{F}_n$ is a ring isomorphism, a routine computation shows that $\mathbf{S} = \mathcal{F}_{n/2}(\widehat{u})_{[0,k)}$. The proof follows from Equation (1). □

[6] This describes the discriminant of $T^2 - \mathrm{Tr}(u)T + N(u)$ whose roots are u and $\sigma(u)$ in $\mathcal{K}_n$. It is then not surprising that $\mathrm{Tr}(\zeta_n^{-1}u)\overline{\mathrm{Tr}(\zeta_n^{-1}u)}$ is a square only in $\mathcal{K}_n$.

Algorithm 5.2 $\mathsf{TowerRecovery}_{\mathcal{F}}(m_0, \ldots, m_{n-1})$

Input: $m_0, \ldots, m_{n-1}$.
Output: $u \in \mathcal{R}_n$ such that u is totally positive and the principal minors of $\mathcal{F}_n(u)$ are m_i's ($0 \leq i < n$).

1: **if** $n = 2$ **then**
2: **return** m_0.
3: **end if**
4: $u^+ \leftarrow \mathsf{TowerRecovery}_{\mathcal{F}}(m_0, \ldots, m_{\frac{n}{2}-1})$ $\{u^+ \text{ is } \mathrm{Tr}(u)/2\}$
5: $\widetilde{u} \leftarrow \mathsf{TowerRecovery}_{\mathcal{F}}(\frac{m_{n/2}}{m_{n/2-1}}, \ldots, \frac{m_{n-1}}{m_{n/2-1}})$ $\{\widetilde{u} \text{ is a conjugate of } \widehat{u} = \frac{2\,\mathrm{N}(u)}{\mathrm{Tr}(u)}\}$
6: Find τ such that $u^+ \cdot \tau(\widetilde{u})$ is a relative norm.
7: $\widehat{u} \leftarrow \tau(\widetilde{u})$
8: $s \leftarrow u^+ \cdot (u^+ - \widehat{u})$
9: $u^- \leftarrow \mathsf{TowerRoot}(\zeta_{n/2}^{-1} s)$ $\{u^- \text{ is a conjugate of } \pm\,\mathrm{Tr}(\zeta_n^{-1}u)/2\}$
10: **return** $u^+ + \zeta_n u^-$

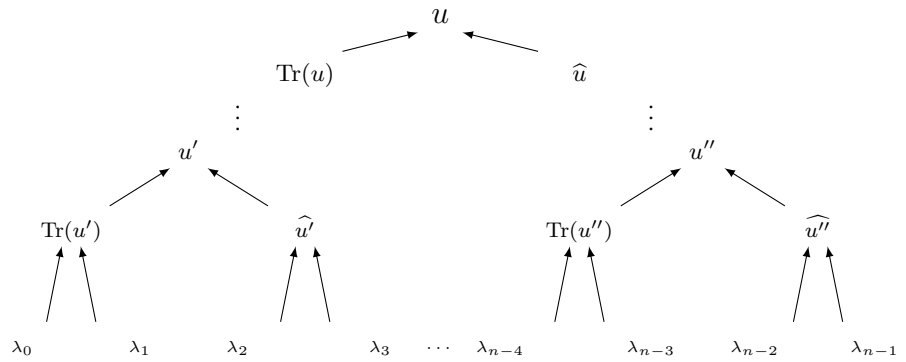

Fig. 1. Binary tree built by $\mathsf{TowerRecovery}_{\mathcal{F}}$.

Lemma 3 tells us that knowing $\mathrm{Tr}(u)$ and the principal minors of $\mathcal{F}_n(u)$ is enough to recover those of $\mathcal{F}_{n/2}(\widehat{u})$, so that the computations in $\mathcal{K}_n$ are again reduced to computing a totally positive element in $\mathcal{K}_{n/2}$ from its minors. Then from Equation (3), we can obtain $\mathrm{Tr}(\zeta_n^{-1}u)\overline{\mathrm{Tr}(\zeta_n^{-1}u)}$. The last step is then to compute a square root of $\zeta_{n/2}^{-1}\,\mathrm{Tr}(\zeta_n^{-1}u)\overline{\mathrm{Tr}(\zeta_n^{-1}u)}$ in $\mathcal{K}_{n/2}$ to recover $\pm\,\mathrm{Tr}(\zeta_n^{-1}u)$. In particular, this step will lead to u or its conjugate $\sigma(u)$. As observed above, this translates ultimately in recovering only a conjugate of u.

Lastly, when $n = 2$, that is, when we work in $\mathbb{Q}(i)$, a totally positive element is in fact in $\mathbb{Q}_+$. This leads to Algorithm 5.2, which is presented in the general context of $\mathcal{K}_n$ to fit the description above, for the sake of simplicity. The algorithm $\mathsf{TowerRoot}$ of Step 9 computes square roots in $\mathcal{K}_n$ and a quasi-quadratic version for integers is presented and analyzed in the next section.

The whole procedure is constructing a binary tree as illustrated in Figure 1. The algorithm can be made to rely essentially only on algebraic integers, which also helps in analyzing its complexity. This is done in Supplementary Material A.2, and gives the claim of Theorem 2. At Step 6, the algorithm finds the (heuristically unique) conjugate $\widehat{u}$ of $\widetilde{u}$ such that $\widehat{u} \cdot u^+$ is a relative norm (since we must have $\widehat{u} \cdot u^+ = \mathrm{N}(u)$ by the above). In practice, in the integral version of this algorithm, we carry out this test not by checking for being a norm, but as an integrality test (see Step 9 in Algorithm A.1).

5.2.1 Computing square roots in cyclotomic towers.

In this section, we will focus on computing square roots of algebraic integers: given $s = t^2 \in \mathcal{R}_n$, compute t. The reason for focusing on

Algorithm 5.3 $\mathsf{TowerRoot}(s)$

Input: $s = t^2$ for some $t \in \mathcal{R}_n$.
Output: $t \in \mathcal{R}_n$.

1: **if** $s \in \mathbb{Z}$ **then**
2: **return** $\mathsf{IntegerSqrt}(s)$
3: **end if**
4: $S \leftarrow \mathrm{N}(s)$ and $S' \leftarrow \mathrm{Tr}(s)$
5: $T \leftarrow \mathsf{TowerRoot}(S)$ $\{T = (-1)^b \mathrm{N}(t)\}$
6: **if** $\mathsf{CheckSqr}(S' + 2T) = \mathsf{False}$ **then**
7: $T \leftarrow -T$
8: **end if**
9: $T^+ \leftarrow \mathsf{TowerRoot}(S' + 2T)$ $\{T^+ = (-1)^{b_0} \mathrm{Tr}(t)\}$
10: $T^- \leftarrow \mathsf{TowerRoot}(\zeta_{n/2}^{-1}(S' - 2T))$ $\{T^- = (-1)^{b_1} \mathrm{Tr}(\zeta_n^{-1} t)\}$
11: **if** $(1/4)(T^+ + \zeta_n T^-)^2 = s$ **then**
12: **return** $(1/2)(T^+ + \zeta_n T^-)$
13: **else**
14: **return** $(1/2)(T^+ - \zeta_n T^-)$
15: **end if**

integers is that both our Algorithm 5.2 and practical applications deal only with algebraic integers. A previous approach was suggested in [26], relying on finding primes with small splitting pattern in $\mathcal{R}_n$, computing square roots in several finite fields and brute-forcing to find the correct candidate. A hassle in analyzing this approach is to first find a prime larger enough than an arbitrary input, and that splits in, say, two factors in $\mathcal{R}_n$. Omitting the cost of finding such a prime, this algorithm can be shown to run in $\widetilde{O}(n^2(\log \|s\|_\infty)^2)$. Our recursive approach does not theoretically rely on finding a correct prime, and again exploits the tower structure to achieve the next claim.

Theorem 3. *Given a square s in $\mathcal{R}_n$, there is a deterministic algorithm that computes $t \in \mathcal{R}_n$ such that $t^2 = s$ in time $\widetilde{O}(n^2 \log \|s\|_\infty)$.*

Recall that the subfield $\mathcal{K}_{n/2}$ is fixed by the automorphism $\sigma(\zeta_n) = -\zeta_n$. For any element t in $\mathcal{R}_n$, recall that $t = \frac{1}{2}(\mathrm{Tr}(t) + \zeta_n \mathrm{Tr}(\zeta_n^{-1} t))$, where Tr is the trace relative to this extension. We can also see that

$$\begin{aligned} \mathrm{Tr}(t)^2 &= \mathrm{Tr}(t^2) + 2\,\mathrm{N}(t) = \mathrm{Tr}(s) + 2\,\mathrm{N}(t), \\ \mathrm{Tr}(\zeta_n^{-1} t)^2 &= \zeta_n^{-2}(\mathrm{Tr}(t^2) - 2\,\mathrm{N}(t)) = \zeta_{n/2}^{-1}(\mathrm{Tr}(s) - 2\,\mathrm{N}(t)), \end{aligned} \tag{4}$$

for the relative norm. Hence recovering $\mathrm{Tr}(t)$ and $\mathrm{Tr}(\zeta_n^{-1} t)$ can be done by computing the square roots of elements in $\mathcal{R}_{n/2}$ determined by s and $\mathrm{N}(t)$. The fact that $\mathrm{N}(s) = \mathrm{N}(t)^2$ leads to Algorithm 5.3.

Notice that square roots are only known up to sign. This means that an algorithm exploiting the tower structure of fields must perform several sign checks to ensure that it will lift the correct root to the next extension. For our algorithm, we only need to check for the sign of $\mathrm{N}(t)$ (the signs of $\mathrm{Tr}(t)$ and $\mathrm{Tr}(\zeta_n^{-1} t)$ can be determined by checking if their current values allow to recover s). This verification happens at Step 6 of Algorithm 5.3, where after computing the square root of $\mathrm{N}(s)$, we know $(-1)^b \mathrm{N}(t)$ for some $b \in \{0, 1\}$. It relies on noticing that from Equations (4), $T_b := \mathrm{Tr}(s) + 2 \cdot (-1)^b \mathrm{N}(t)$ is a square in $\mathcal{K}_{n/2}$ if and only if $b = 0$, in which case $T_b = \mathrm{Tr}(t)^2$. (Else, $\zeta_n^{-2} T_b$ is the square $\mathrm{Tr}(\zeta_n^{-1} t)^2$ in $\mathcal{K}_{n/2}$.) We show in Supplementary Material A.3 that this observation can be extended to a sign check that runs in $\widetilde{O}(n \cdot \log \|s\|_\infty)$.

In practice, we can use the following approach: since n is small, we can easily precompute a prime integer p such that $p - 1 \equiv n \bmod 2n$. For such a prime, there is a primitive n^{th} root ω of unity in $\mathbb{F}_p$, and such a root cannot be a square in $\mathbb{F}_p$ (else $2n$ would divide $p-1$). Checking squareness then amounts to checking which of $T_b(\omega)$ or $\omega^{-2}T_b(\omega)$ is a square $\bmod p$ by computing a Legendre symbol. While we need such primes for any power of 2 that is smaller than n, in any case, this checks is done in quasi-linear time. Compared to [26], the size of p here does not matter.

Let us denote by $\mathsf{SQRT}(n, S)$ the complexity of Algorithm 5.3 for an input $s \in \mathcal{R}_n$ with coefficients of size $S = \log \|s\|_\infty$. Using e.g. FFT based multiplication of polynomials, $\mathrm{N}(s)$ can be computed in $\widetilde{O}(nS)$, and has bitsize at most $2S + \log n$. Recall that the so-called canonical embedding of any $s \in \mathcal{K}_n$ is the vector $\tau(s)$ of its evaluations at the roots of x^n+1. It is well-known that it satisfies $\|\tau(s)\| = \sqrt{n}\|s\|$, so that $\|\tau(s)\|_\infty \leq n\|s\|_\infty$ by norm equivalence. If $s = t^2$ we see that $\|\tau(s)\|_\infty = \|\tau(t)\|_\infty^2$. Using again norm equivalence, we obtain $\|t\|_\infty \leq \sqrt{n}\|s\|_\infty^{1/2}$. In the case of $\mathrm{N}(s) = \mathrm{N}(t)^2$, we obtain that $\mathrm{N}(t)$ has size at most $S + \log n$. The cost of CheckSqr is at most $\widetilde{O}(nS)$, so we obtain

$$\mathsf{SQRT}(n, S) = \mathsf{SQRT}\left(\frac{n}{2}, 2S + \log n\right) + 2\mathsf{SQRT}\left(\frac{n}{2}, S + \log n\right) + \widetilde{O}(nS). \tag{5}$$

A tedious computation (see Supplementary Material A.4) gives us Theorem 3.

6 Side-Channel Leakage of the Gram–Schmidt Norms

Our algorithms in Section 5 rely on the knowledge of the exact Gram-Schmidt norms $\|\mathbf{b}_i^*\|$. In this section, we show that in the original implementations of DLP and FALCON, approximations of $\|\mathbf{b}_i^*\|$'s can be obtained by exploiting the leakage induced by a non constant-time rejection sampling.

In previous works targeting the rejection phase, the standard deviation of the sampler was a public constant. This work deals with a different situation, as both the centers and the standard deviations used by the samplers of DLP and FALCON are secret values determined by the secret key. These samplers output Gaussian vectors by relying on an integer Gaussian sampler, which performs rejection sampling. The secret standard deviation for the i^{th} integer Gaussian is computed as $\sigma_i = \sigma/\|\mathbf{b}_i^*\|$ for some fixed σ, so that exposure of the σ_i's means the exposure of the Gram-Schmidt norms. The idea of the attack stems from the simple observation that the acceptance rate of the sampler is essentially a linear function of its current σ_i. In this section, we show how, by a timing attack, one may recover all acceptance rates from sufficiently many signatures by computing a well-chosen maximum likelihood estimator. Recovering approximations of the $\|\mathbf{b}_i^*\|$'s then follows straightforwardly.

6.1 Leakage in the DLP Scheme

We first target the Gaussian sampling in the original implementation [47], described in Algorithms 6.1 and 6.2. It samples "shifted" Gaussian integers by relying on three layers of Gaussian integer sampling with rejection. More precisely, the target Gaussian distribution at the "top" layer has a center which depends on secret data and varies during each call. To deal with the varying center, the "shifted" sample is generated by combining zero-centered sampler and rejection sampling. Yet the zero-centered sampler has the same standard deviation as the "shifted" one, and

Algorithm 6.1 DLP base sampler $\mathsf{DLPIntSampler}(\sigma, c)$

Input: $c \in [0, 1)$ and $\sigma \geq \eta_\epsilon(\mathbb{Z})$.
Output: $z \in \mathbb{Z}$ following $D_{\mathbb{Z},\sigma,c}$.
1: $z \leftarrow \mathsf{DLPCenteredIntSampler}(\sigma)$
2: $b \leftarrow U(\{0, 1\})$
3: $z \leftarrow z + b$
4: return z with probability $\frac{\rho_{\sigma,c}(z)}{\rho_\sigma(z)+\rho_\sigma(z-1)}$, otherwise restart.

Algorithm 6.2 DLP centered base sampler $\mathsf{DLPCenteredIntSampler}(\sigma)$

Input: $\sigma \geq \eta_\epsilon(\mathbb{Z})$.
Output: $z \in \mathbb{Z}$ following $D_{\mathbb{Z},\sigma}$.
1: $k \leftarrow \lceil \frac{\sigma}{\widehat{\sigma}} \rceil$ where $\widehat{\sigma} = \sqrt{\frac{1}{2\log(2)}}$
2: $z \leftarrow \mathsf{IntSampler}(k\widehat{\sigma})$
3: return z with probability $\frac{\rho_\sigma(z)}{\rho_{k\widehat{\sigma}}(z)}$, otherwise restart.

the standard deviation depends on the secret key. At the "intermediate" layer, also by rejection sampling, the sampler rectifies a public zero-centered sample to a secret-dependent one.

At the "bottom" layer, the algorithm IntSampler actually follows the BLISS sampler [11] that is already subject to side-channel attacks [10, 44, 16]. We stress again that our attack does not target this algorithm, so that the reader can assume a constant-time version of it is actually used here. The weakness we are exploiting is a non constant-time implementation of Algorithm 6.2 in the "intermediate" layer. We now describe how to actually approximate the σ_i's using this leakage.

Let $k_i = \lceil \frac{\sigma_i}{\widehat{\sigma}} \rceil$. It can be verified that the average acceptance probability of Algorithm 6.2 is $AR(\sigma_i) = \frac{\rho_{\sigma_i}(\mathbb{Z})}{\rho_{k\widehat{\sigma}}(\mathbb{Z})}$. As required by the KGPV algorithm, we know that $k_i\widehat{\sigma} \geq \sigma_i \geq \eta'_\epsilon(\mathbb{Z})$ and by Corollary 1 we have $AR(\sigma_i) \in \frac{\sigma_i}{k_i\widehat{\sigma}} \cdot \left[\frac{1-\epsilon}{1+\epsilon}, 1\right]$. Since ϵ is very small in this context, we do not lose much by assuming that $AR(\sigma_i) = \frac{\sigma_i}{k_i\widehat{\sigma}}$.

Next, for a given σ_i, the number of trials before Algorithm 6.2 outputs its result follows a geometric distribution $\mathrm{Geo}_{AR(\sigma_i)}$. We let $\overline{AR}_i$ be maximum likelihood estimators for the $AR(\sigma_i)$'s associated to N executions of the KGPV sampler, that we compute using Lemma 1. We now want to determine the k_i's to compute $\overline{\sigma_i} = k_i\hat{\sigma}\overline{AR}_i$. Concretely, for the suggested parameters, we can set $k_i = 3$ for all i at the beginning and measure $\overline{AR}_i$. Because the first half of the σ_i's are in a small interval and increase slowly, it may be the case at some step that $\overline{AR}_{i+1}$ is significantly smaller than $\overline{AR}_i$ (say, $1.1 \cdot \overline{AR}_{i+1} < \overline{AR}_i$). This means that $k_{i+1} = k_i + 1$, and we then increase by one all the next k_i's. This approach can be done until $\overline{AR}_n$ is obtained, and works well in practice. Lastly, Lemma 1 tells us that for large enough α and p, taking $N \geq 2^{2(p+\log\alpha)}$ implies $|\overline{\sigma}_i - \sigma_i| \leq 2^{-p} \cdot \sigma_i$ for all $0 \leq i < 2n$ with high probability.

From [13], the constant σ is publicly known. This allows us to have approximations $\overline{b_i} = \frac{\sigma}{\overline{\sigma_i}}$, which we then expect are up to p bits of accuracy on $\|\mathbf{b}_i^*\|$.

6.2 Leakage in the FALCON Scheme

We now describe how the original implementation of FALCON presents a similar leakage of Gram–Schmidt norms via timing side-channels. In contrast to the previous section, the integer sampler of FALCON is based on one public half-Gaussian sampler and some rejection sampling to reflect sensitive standard deviations and centers. The procedure is shown in Algorithm 6.3.

Algorithm 6.3 FALCON base sampler $\mathsf{FalconIntSampler}(\sigma, c)$

Input: $c \in [0,1)$ and $\sigma \geq \eta_\epsilon(\mathbb{Z})$.
Output: $z' \in \mathbb{Z}$ following $D_{\mathbb{Z},\sigma,c}$.
1: $z \hookleftarrow D^+_{\mathbb{Z},\widehat{\sigma}}$ where $\widehat{\sigma} = 2$
2: $b \hookleftarrow U(\{0,1\})$
3: return $z' = b + (2b-1)z$ with probability $\exp\left(\frac{z^2}{2\widehat{\sigma}^2} - \frac{(b+(2b-1)z-c)^2}{2\sigma^2}\right)$, otherwise restart.

Our analysis does not target the "half-Gaussian" sampler $D^+_{\mathbb{Z},\widehat{\sigma}}$, so that we omit its description. It can be implemented in a constant-time way [30], but this has no bearing on the leakage we describe.

We first consider c_i and σ_i to be fixed. Following Algorithm 6.3, we let $p(z,b) = \exp\left(\frac{z^2}{2\widehat{\sigma}^2} - \frac{(b+(2b-1)z-c_i)^2}{2\sigma_i^2}\right)$ be the acceptance probability and note that

$$p(z,0) = \frac{1}{\rho_{\hat{\sigma}}(z)} \exp\left(-\frac{(-z-c)^2}{2\sigma_i^2}\right) \quad \text{and} \quad p(z,1) = \frac{1}{\rho_{\hat{\sigma}}(z)} \exp\left(-\frac{(z+1-c)^2}{2\sigma_i^2}\right).$$

Then the average acceptance probability for fixed c and σ_i satisfies

$$\begin{aligned}\mathbb{E}_{z,b}[p(z,b)] =& \frac{1}{2\rho_{\hat{\sigma}}(\mathbb{N})} \sum_{z\in\mathbb{N}} \left(\exp\left(-\frac{(-z-c)^2}{2\sigma_i^2}\right) + \exp\left(-\frac{(z+1-c)^2}{2\sigma_i^2}\right)\right) \\ =& \frac{\rho_{\sigma_i}(\mathbb{Z}-c)}{2\rho_{\hat{\sigma}}(\mathbb{N})}.\end{aligned}$$

As $\widehat{\sigma} \geq \sigma_i \geq \eta'_\epsilon(\mathbb{Z})$ for a very small ϵ, we can again use Lemma 2 to have that $\rho_{\sigma_i}(\mathbb{Z}-c) \approx \rho_{\sigma_i}(\mathbb{Z})$. This allows us to consider the average acceptance probability as a function $AR(\sigma_i)$, independent of c. Using that $2\rho^+_{\hat{\sigma}}(\mathbb{N}) = \rho_{\hat{\sigma}}(\mathbb{Z}) + 1$ combined with Corollary 1, we write $AR(\sigma_i) = \frac{\sigma_i\sqrt{2\pi}}{1+2\sqrt{2\pi}}$. Then an application of Lemma 1 gives the needed number of traces to approximate σ_i up to a desired accuracy.

7 Practical Attack Against the DLP Scheme

For the methods in Section 6, measure errors seem inevitable in practice. To mount a practical attack, we have to take into account this point. In this section, we show that it is feasible to compute a totally positive element even with noisy diagonal coefficients of its LDL decomposition.

First we adapt the algorithm $\mathsf{Recovery}_{\mathcal{A}}$ (Algorithm 5.1) to the noisy input in Section 7.1. To determine each coefficient, we need to solve a quadratic inequality instead of an equation due to the noise. As a consequence, each quadratic inequality may lead to several candidates of the coefficient. According to if there is a candidate or not, the algorithm extends prefixes hopefully extending to a valid solution or eliminates wrong prefixes. Thus the algorithm behaves as a tree search.

Then we detail in Section 7.2 some implementation techniques to accelerate the recovery algorithm in the context of the DLP scheme. While the algorithm is easy to follow, adapting it to practical noisy case is not trivial.

At last, we report experimental results in Section 7.3. As a conclusion, given the full timing leakage of about 2^{34} signatures, one may practically break the DLP parameter claimed for 192-bit security with a good chance. We bring some theoretical support for this value in Section 7.4.

Algorithm 7.1 $\mathsf{Recovery}_{\mathcal{A}}(\delta, \{\overline{d_i}\}_i, \mathsf{prefix})$

Input: $\delta \in \left[0, \frac{1}{2}\right)$, $\mathsf{prefix} = \left(\overline{A_0}, \cdots, \overline{A_{l-1}}\right) \in \mathbb{Z}^l$ and for all i
$\overline{d_i} = d_i + \epsilon_i$ where $d_i = \det\left(\mathcal{A}_n(A)_{[0,i]}\right) / \det\left(\mathcal{A}_n(A)_{[0,i-1]}\right)$ and $|\epsilon_i| \leq \delta$.
Output: a list of candidates of A in which each candidate A'
(1) takes prefix as the first l coefficients;
(2) satisfies $|\overline{d_i} - d'_i| < \delta$ where $d'_i = \det\left(\mathcal{A}_n(A')_{[0,i]}\right) / \det\left(\mathcal{A}_n(A')_{[0,i-1]}\right)$.

$S \leftarrow \emptyset$
if $l = \frac{n}{2}$ **then**
  $S \leftarrow \{\overline{A_0} + \sum_{i=1}^{\frac{n}{2}-1} \overline{A_i}(X^i + X^{-i})\}$
else
  $\mathbf{T} \leftarrow (\overline{A_{|i-j|}})_{i,j\in[0,l)}$, $\mathbf{t} \leftarrow \left(0, \overline{A_{l-1}}, \cdots, \overline{A_1}\right)$
  $Q_a \leftarrow \mathbf{T}_{0,0}^{-1}$, $Q_b \leftarrow \sum_{i=1}^{l-1} \mathbf{T}_{0,i}^{-1} t_i$, $Q_c \leftarrow \mathbf{t}^t\mathbf{T}^{-1}\mathbf{t} - \overline{A_0} + \overline{d_l}$
  $S_l \leftarrow \{x \in \mathbb{Z} : |Q_a x^2 + 2Q_b x + Q_c| < \delta\}$ {all possible A_l}
  for $a \in S_l$ **do**
    $\mathsf{prefix}' \leftarrow (\mathsf{prefix}, a) \in \mathbb{Z}^{l+1}$
    $S \leftarrow S \bigcup \mathsf{Recovery}_{\mathcal{A}}(\delta, \{\overline{d_i}\}_i, \mathsf{prefix}')$
  end for
end if
return S

7.1 Totally Positive Recovery With Noisy Inputs

Section 5.1 has sketched the exact recovery algorithm. To tackle the measure errors, we introduce a new parameter to denote the error bound. The modified algorithm proceeds in the same way: given a prefix $(A_0, \cdots, A_{l-1})$, it computes all possible A_l's satisfying the error bound condition and extends or eliminates the prefix according to if it can lead to a valid solution. A formal algorithmic description is provided in Algorithm 7.1. For convenience, we use the (noisy) diagonal coefficients (i.e. secret Gram-Schmidt norms) of the LDL decomposition as input. In fact, Proposition 1 has shown the equivalence between the diagonal part and principal minors. In addition, we include prefix in the input for ease of description. The initial prefix is $\mathsf{prefix} = \overline{A_0} = \lfloor \overline{d_0} \rceil$. Clearly, the correct A must be in the final candidate list.

7.2 Practical Tweaks in the DLP Setting

Aiming at the DLP signature, we implemented our side-channel attack. By the following techniques, one can boost the practical performance of the recovery algorithm significantly and reduce the number of required signatures.

Fast computation of the quadratic equation. Exploiting the Toeplitz structure of $\mathcal{A}_n(A)$, we propose a fast algorithm to compute the quadratic equation, i.e. (Q_a, Q_b, Q_c), that requires only $O(l)$ multiplications and additions. The idea is as follows. Let $\mathbf{T}_i = \mathcal{A}_n(A)_{[0,i]}$. Let $\mathbf{u}_i = (A_1, \cdots, A_i)$ and $\mathbf{v}_i = (A_i, \cdots, A_1)$, then

$$\mathbf{T}_i = \begin{pmatrix} \mathbf{T}_{i-1} & \mathbf{v}_i^t \\ \mathbf{v}_i & A_0 \end{pmatrix} = \begin{pmatrix} A_0 & \mathbf{u}_i \\ \mathbf{u}_i^t & \mathbf{T}_{i-1} \end{pmatrix}.$$

Let $\mathbf{r}_i = \mathbf{v}_i\mathbf{T}_{i-1}^{-1}$, $\mathbf{s}_i = \mathbf{u}_i\mathbf{T}_{i-1}^{-1}$ which is the reverse of $\mathbf{r}_i$ and $d_i = A_0 - \langle \mathbf{v}_i, \mathbf{r}_i \rangle = A_0 - \langle \mathbf{u}_i, \mathbf{s}_i \rangle$. A straightforward computation leads to that

$$\mathbf{T}_i^{-1} = \begin{pmatrix} \mathbf{T}_{i-1}^{-1} + \mathbf{r}_i^t\mathbf{r}_i/d_i & -\mathbf{r}_i^t/d_i \\ -\mathbf{r}_i/d_i & 1/d_i \end{pmatrix}.$$

Let $f_i = \langle \mathbf{r}_i, \mathbf{u}_i \rangle = \langle \mathbf{s}_i, \mathbf{v}_i \rangle$, then the quadratic equation of A_i is

$$d_i = A_0 - \langle \mathbf{v}_i, \mathbf{r}_i \rangle = A_0 - (A_i - f_{i-1})^2/d_{i-1} - \langle \mathbf{v}_{i-1}, \mathbf{r}_{i-1} \rangle.$$

Remark that d_i is the square of the last Gram-Schmidt norm. Because $\overline{d_i}$, a noisy d_i, is the input, combining $f_{i-1}, \mathbf{v}_{i-1}, \mathbf{r}_{i-1}$ would determine all possible A_i's. Once A_i is recovered, one can then compute $\mathbf{r}_i$, $\mathbf{s}_i$ according to

$$\mathbf{s}_i = \left(-\tfrac{A_i - f_{i-1}}{d_{i-1}} \mathbf{r}_{i-1} + \mathbf{s}_{i-1}, \ \tfrac{A_i - f_{i-1}}{d_{i-1}} \right)$$

and further compute d_i, f_i. As the recovery algorithm starts with $i = 1$ (i.e. prefix $= A_0$), we can compute the sequences $\{d_i\}, \{f_i\}, \{\mathbf{r}_i\}, \{\mathbf{s}_i\}$ on the fly.

Remark 1. The input matrix is very well conditioned, so we can use a precision of only $O(\log n)$ bits.

Remark 2. The above method implies an algorithm of complexity $\widetilde{O}(n^2)$ for the exact case (Section 5.1).

Pruning. We expect that when a mistake is made in the prefix, the error committed in the Gram-Schmidt will be larger. We therefore propose to prune prefixes when $\sum_{k=i}^{j} e_k^2/\sigma_k^2 \geq B_{j-i}$ for some i, j where e_k is the difference between the measured k-th squared Gram-Schmidt norm and the one of the prefix. The bound B_l is selected so that for e_k a Gaussian of standard deviation σ_k, the condition is verified except with probability $\tau/\sqrt{l}$. The failure probability τ is geometrically decreased until the correct solution is found.

Verifying candidates. Let $A = f\overline{f} + g\overline{g}$, then $f\overline{f} = A(1 + h\overline{h}) \bmod q$. As mentioned in Section 4, all coefficients except the constant one of $f\overline{f}$ would be much smaller the modulus q. This can be used to check if a candidate is correct. In addition, both $A(x)$ and $A(-x)$ are the final candidates, we also check $A(1 + h(-x)\overline{h}(-x))$ to ensure that the correct $A(-x)$ will not to be eliminated. Once either $A(x)$ or $A(-x)$ is found, we terminate the algorithm.

The use of symplecticity. As observed in [19], the trapdoor basis $\mathbf{B}_{f,g}$ is q-symplectic and thus $\|\mathbf{b}_i^*\| \cdot \|\mathbf{b}_{2n-1-i}^*\| = q$. Based on that, we combine the samples of the i-th and $(2n-1-i)$-th Gaussians to approximate $\|\mathbf{b}_i^*\|$. This helps to refine the approximations and thus to reduce the number of signatures enabling a practical attack.

7.3 Experimental Results

We validate the recovery algorithm on practical DLP instances. The timing leakage we exploited comes from centered Gaussian samplings (Algorithm 6.2) that only depends on the secret key itself not the hashed message. Hence, instead of executing complete signing, we only perform centered Gaussian samplings. We mean by *sample size* the number of collected Gaussian samples. In fact, considering the rejection sampling in Algorithm 6.1, one requires $N/2$ signatures to generate N samples per centered Gaussian.

We tested our algorithm on ten instances, and result is shown in Table 1.

In one instance, the recovery algorithm found millions of candidate solutions with Gram-Schmidt norms closer to the noisy ones than the correct solution, in the sense that they had a larger τ. This indicates that the recovery algorithm is relatively close to optimality.

Table 1. Experimental validation of the recovery of $f\overline{f} + g\overline{g}$. The first column and row indicate the time limit and the logarithm of used sample size respectively. The remaining data shows how many instances out of 10 are solved correctly within the time limit and with given number of samples.

	36.5	36.0	35.5	35.0	34.5	34.0
< 1 s	8	7	4	3	0	0
< 10 s	9	8	6	4	1	0
$< 10^2$ s	10	9	7	4	3	1
$< 10^3$ s	10	10	8	4	4	1
$< 10^4$ s	10	10	8	5	4	1
$< 10^5$ s	10	10	8	6	4	2
$< 5 \cdot 10^5$ s	10	10	9	7	4	3

7.4 Precision Required on the Gram–Schmidt Norms

We try here to give a closed formula for the number of samples needed. We recall that the relative error with respect to the Gram-Schmidt norm (squared) is $\Theta(1/\sqrt{N})$ where N is the number of samples.

A fast recovery corresponds to the case where only one root is close to an integer; and in particular increasing by one the new coefficient must change by $\Omega(1/\sqrt{N})$ the Gram-Schmidt norm. This is not an equivalence because there is another root to the quadratic form, but we will assume this is enough.

Let b_1 be the first row of $\big(\mathcal{A}_n(f)\ \mathcal{A}_n(g)\big)$, and b_i the i-th row for $i \geq 2$. We define pb_i as the projection of b_1 orthogonally to $b_2, \ldots, b_{i-1}$. We expect that $\|pb_i\| \approx \sqrt{\frac{2n-i+2}{2n}}\|b_1\|$. Consider the Gram matrix of the family $b_1, \ldots, b_{i-1}, b_i \pm \frac{pb}{\|b_1\|^2}$. We have indeed changed only the top right/bottom left coefficients by ± 1, beside the bottom right coordinate. Clearly this does not change the i-th Gram-Schmidt vector; so the absolute change in the i-th Gram-Schmidt norm squared is

$$\left\| b_i \pm \frac{pb_i}{\|b_1\|^2} \right\|^2 - \|b_i\|^2 \approx \pm \frac{\langle b_i, pb_i \rangle}{\|b_1\|^2}.$$

The Gram-Schmidt norm squared is roughly $\|pb_i\|^2$.

Getting only one solution at each step with constant probability corresponds to

$$\langle b_i, pb_i \rangle \geq \frac{\|b_i\|\|pb_i\|}{\sqrt{2n-i+2}}$$

(assuming the scalar product is distributed as a Gaussian) which means a total number of samples of

$$N = \Theta\left(\frac{\sqrt{2n-i+2}\|pb_i\|\|b_1\|^2}{\|b_i\|\|pb_i\|} \right)^2 = \Theta(n\|b_1\|^2) = \Theta(nq^2).$$

This gives roughly 2^{29} samples, which is similar to what the search algorithm requires.

Getting only one solution at each step with probability $1 - 1/n$ corresponds to

$$\langle b_i, pb_i \rangle \geq \frac{\|b_i\|\|pb_i\|}{n\sqrt{2n-i+2}}$$

and $N = \Theta(n^3 q^2)$. This would be 2^{57} samples.

8 Conclusion and Future Work

In this paper, we have investigated the side-channel security of the two main efficient hash-and-sign lattice-based signature schemes: DLP and FALCON (focusing on their original implementations, although our results carry over to several later implementations as well). The two main takeaways of our analysis are that:

1. the Gram–Schmidt norms of the secret basis leak through timing side-channels; and
2. knowing the Gram–Schmidt norms allows to fully recover the secret key.

Interestingly, however, there is a slight mismatch between those two results: the side-channel leakage only provides *approximate* values of the Gram–Schmidt norms, whereas secret key recovery a priori requires *exact* values. We are able to bridge this gap in the case of DLP by combining the recovery algorithm with a pruned tree search. This lets us mount a concrete attack against DLP that recovers the key from 2^{33} to 2^{35} DLP traces in practice for the high security parameters of DLP (claiming 192 bits of security).

However, the gap remains in the case of FALCON: we do not know how to modify our recovery algorithm so as to deal with approximate inputs, and as a result apply it to a concrete attack. This is left as a challenging open problem for future work.

Also left for future work on the more theoretical side is the problem of giving an intrinsic description of our recovery algorithms in terms of algebraic quantities associated with the corresponding totally positive elements (or equivalently, to give an algebraic interpretation of the LDL decomposition for algebraically structured self-adjoint matrices). In particular, in the FALCON case, our approach shows that the Gram–Schmidt norms characterize the Galois conjugacy class of a totally positive element. This strongly suggests that they should admit a nice algebraic description, but it remains elusive for now.

On a positive note, we finally recall that the problem of finding countermeasures against the leakage discussed in this paper is fortunately *already solved*, thanks to the recent work of Prest, Ricosset and Rossi [49]. And that countermeasure has very recently been implemented into FALCON [46], so the leak can be considered as patched! The overhead of that countermeasure is modest in the case of FALCON, thanks to the small range in which the possible standard deviations occur; however, it could become more costly for samplers that need to accommodate a wider range of standard deviations.

An alternate possible countermeasure could be to use Peikert's convolution sampling [43] in preference to the KGPV approach, as it eliminates the need for varying standard deviations, and is easier to implement even without floating point arithmetic. It does have the drawback of sampling wider Gaussians, however, and hence leads to less compact parameter choices.

References

[1] Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986)

[2] Bai, S., Galbraith, S.D.: An improved compression technique for signatures based on learning with errors. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 28–47. Springer, Heidelberg (Feb 2014)

[3] Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen 296(1), 625–635 (1993)

[4] Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Grégoire, B., Rossi, M., Tibouchi, M.: Masking the GLP lattice-based signature scheme at any order. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 354–384. Springer, Heidelberg (Apr / May 2018)

[5] Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Rossi, M., Tibouchi, M.: GALACTICS: Gaussian Sampling for Lattice-Based Constant-Time Implementation of Cryptographic Signatures, Revisited. Cryptology ePrint Archive, Report 2019/511 (2019)

[6] Bindel, N., Akleylek, S., Alkim, E., Barreto, P.S.L.M., Buchmann, J., Eaton, E., Gutoski, G., Kramer, J., Longa, P., Polat, H., Ricardini, J.E., Zanon, G.: qTESLA. Tech. rep., National Institute of Standards and Technology (2019), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions

[7] Bootle, J., Delaplace, C., Espitau, T., Fouque, P.A., Tibouchi, M.: LWE without modular reduction and improved side-channel attacks against BLISS. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 494–524. Springer, Heidelberg (Dec 2018)

[8] Bostan, A.: Algorithmes rapides pour les polynômes, séries formelles et matrices (2010), https://specfun.inria.fr/bostan/publications/Bostan10.pdf

[9] Brent, R.P., Gustavson, F.G., Yun, D.Y.: Fast solution of toeplitz systems of equations and computation of padé approximants. Journal of Algorithms 1(3), 259 – 295 (1980)

[10] Bruinderink, L.G., Hülsing, A., Lange, T., Yarom, Y.: Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 323–345. Springer, Heidelberg (Aug 2016)

[11] Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (Aug 2013)

[12] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES 2018(1), 238–268 (2018), https://tches.iacr.org/index.php/TCHES/article/view/839

[13] Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over NTRU lattices. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 22–41. Springer, Heidelberg (Dec 2014)

[14] Ducas, L., Nguyen, P.Q.: Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 433–450. Springer, Heidelberg (Dec 2012)

[15] Ducas, L., Prest, T.: Fast Fourier Orthogonalization. In: ISSAC. pp. 191–198 (2016)

[16] Espitau, T., Fouque, P.A., Gérard, B., Tibouchi, M.: Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1857–1874. ACM Press (Oct / Nov 2017)

[17] Espitau, T., Fouque, P., Gérard, B., Tibouchi, M.: Loop-abort faults on lattice-based signature schemes and key exchange protocols. IEEE Trans. Computers 67(11), 1535–1549 (2018), https://doi.org/10.1109/TC.2018.2833119

[18] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987)

[19] Gama, N., Howgrave-Graham, N., Nguyen, P.Q.: Symplectic lattice reduction and NTRU. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 233–253. Springer, Heidelberg (May / Jun 2006)
[20] Gentry, C., Jonsson, J., Stern, J., Szydlo, M.: Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 1–20. Springer, Heidelberg (Dec 2001)
[21] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008)
[22] Gentry, C., Szydlo, M.: Cryptanalysis of the revised NTRU signature scheme. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 299–320. Springer, Heidelberg (Apr / May 2002)
[23] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (Aug 1997)
[24] Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: A signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (Sep 2012)
[25] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSIGN: Digital signatures using the NTRU lattice. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 122–140. Springer, Heidelberg (Apr 2003)
[26] Hoffstein, J., Lieman, D., Silverman, J.H.: Polynomial rings and efficient public key authentication (1999)
[27] Hogg, R.V., McKean, J.W., Craig, A.T.: Introduction to Mathematical Satistics (8th edition). Pearson (2018)
[28] Hülsing, A., Lange, T., Smeets, K.: Rounded gaussians - fast and secure constant-time sampling for lattice-based crypto. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 728–757. Springer, Heidelberg (Mar 2018)
[29] Karmakar, A., Roy, S.S., Reparaz, O., Vercauteren, F., Verbauwhede, I.: Constant-Time Discrete Gaussian Sampling. IEEE Transactions on Computers 67(11), 1561–1571 (2018)
[30] Karmakar, A., Roy, S.S., Vercauteren, F., Verbauwhede, I.: Pushing the speed limit of constant-time discrete Gaussian sampling. A case study on the Falcon signature scheme. In: DAC 2019 (2019)
[31] Klein, P.N.: Finding the closest lattice vector when it's unusually close. In: Shmoys, D.B. (ed.) 11th SODA. pp. 937–941. ACM-SIAM (Jan 2000)
[32] Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (Dec 2009)
[33] Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012)
[34] Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2019), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
[35] Lyubashevsky, V., Prest, T.: Quadratic time, linear space algorithms for Gram-Schmidt orthogonalization and Gaussian sampling in structured lattices. In: Oswald, E., Fischlin, M.

(eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 789–815. Springer, Heidelberg (Apr 2015)
[36] McCarthy, S., Howe, J., Smyth, N., Brannigan, S., O'Neill, M.: BEARZ attack FALCON: implementation attacks with countermeasures on the FALCON signature scheme. In: Obaidat, M.S., Samarati, P. (eds.) SECRYPT. pp. 61–71 (2019)
[37] McCarthy, S., Smyth, N., O'Sullivan, E.: A practical implementation of identity-based encryption over NTRU lattices. In: O'Neill, M. (ed.) 16th IMA International Conference on Cryptography and Coding. LNCS, vol. 10655, pp. 227–246. Springer, Heidelberg (Dec 2017)
[38] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (Apr 2012)
[39] Micciancio, D., Regev, O.: Worst-Case to Average-Case Reductions Based on Gaussian Measures. SIAM Journal on Computing 37(1), 267–302 (2007)
[40] Micciancio, D., Walter, M.: Gaussian sampling over the integers: Efficient, generic, constant-time. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 455–485. Springer, Heidelberg (Aug 2017)
[41] Nguyen, P.Q., Regev, O.: Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 271–288. Springer, Heidelberg (May / Jun 2006)
[42] Oder, T., Speith, J., Höltgen, K., Güneysu, T.: Towards practical microcontroller implementation of the signature scheme Falcon. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography - 10th International Conference, PQCrypto 2018. pp. 65–80. Springer, Heidelberg (2019)
[43] Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (Aug 2010)
[44] Pessl, P., Bruinderink, L.G., Yarom, Y.: To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1843–1855. ACM Press (Oct / Nov 2017)
[45] Plantard, T., Sipasseuth, A., Dumondelle, C., Susilo, W.: DRS. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
[46] Pornin, T.: New Efficient, Constant-Time Implementations of Falcon (Aug 2019), https://falcon-sign.info/falcon-impl-20190802.pdf
[47] Prest, T.: Proof-of-concept implementation of an identity-based encryption scheme over NTRU lattices (2014), https://github.com/tprest/Lattice-IBE
[48] Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2019), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
[49] Prest, T., Ricosset, T., Rossi, M.: Simple, Fast and Constant-Time Gaussian Sampling over the Integers for Falcon. In: Second PQC Standardization Conference (2019)
[50] Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (May 2011)
[51] Tibouchi, M., Wallet, A.: One bit is all it takes: a devastating timing attack on BLISS's non-constant time sign flips. Cryptology ePrint Archive, Report 2019/898 (2019), https://eprint.iacr.org/2019/898

[52] Yu, Y., Ducas, L.: Learning strikes again: The case of the DRS signature scheme. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 525–543. Springer, Heidelberg (Dec 2018)

[53] Zhang, Z., Chen, C., Hoffstein, J., Whyte, W.: pqNTRUSign. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

[54] Zhao, R.K., Steinfeld, R., Sakzad, A.: FACCT: FAst, Compact, and Constant-Time Discrete Gaussian Sampler over Integers. IACR Cryptology ePrint Archive (2018), report 2018/1234

Supplementary Material

A Complexity Analysis of the Algorithms

A.1 Complexity of Recovery$_{\mathcal{A}}$

We analyze first the size of the quantities involved in Algorithm 5.1, and let $dA = B$ for $d \in \mathbb{Z} \setminus \{0\}$ and $B \in \mathcal{R}_n$. By construction, we have $\mathbf{w}_i = \mathcal{A}_n(A)^{-1}_{[0,i-1]} \cdot \mathbf{v}_i^t$. We observe that $\mathcal{A}_n(A)^{-1}_{[0,i]} = \mathrm{diag}(d)_i \cdot \mathcal{A}_n(B)^{-1}_{[0,i]}$ holds for all $i \leq n$. Next we know that $\mathcal{A}_n\big(B\big)^{-1}_{[0,i]} = \det(\mathcal{A}_n(B)_{[0,i]})^{-1} \cdot \mathrm{adj}\big(\mathcal{A}_n(A)_{[0,i]}\big)^t$, where adj means taking the adjugate matrix. Using Hadamard's inequality, all the minors of $\mathcal{A}_n\big(B\big)_{[0,i]}$ are bounded by $\|B\|_\infty^{i-1} \cdot (i-1)^{(i-1)/2}$, and $|\det \mathcal{A}_n(B)_{[0,i]}| \leq \|B\|_\infty^i i^{i/2}$. Therefore we obtain $\|\mathcal{A}_n(A)^{-1}_{[0,i]}\|_{\max} \leq (|d| \cdot i \cdot \|B\|_\infty^2)^i$ for all $i \leq n$.

Now, we have $\|\mathbf{w}_i\|_\infty = \|\mathcal{A}_n(A)^{-1}_{[0,i-1]}\mathbf{v}_i^t\|_\infty \leq i \cdot \|\mathcal{A}_n(A)^{-1}_{[0,i-1]}\|_{\max} \cdot \|\mathbf{v}_i\|_\infty$. Let $S = \log|d| + \log\|B\|_\infty$. By definition, we see that $\log\|\mathbf{v}_i\|_\infty \leq S$, so that we can write $\log\|\mathbf{w}_i\|_\infty \leq (2i+1)S + (i+1)\log i$. This also shows that $\mathbf{v}_i\mathbf{w}_i^t$ is a quadratic polynomial with coefficients of size at most $2(i+1)(S+\log i)$. The rational $\frac{m_i}{m_{i-1}}$ has size at most $2i(\log\|B\|_\infty + \log i) + \log|d|$. This means that the quadratic equation to solve at step i has coefficients of size $O(i(S+\log i))$. A bound on the size of any quantity involved in Algorithm 5.1 during the i^{th} is then $\widetilde{O}(iS)$.

The main computational tasks are solving the linear equations of Step 6 and the computation of a square root in $\mathbb{Q}$. Observe that at step i, $\mathcal{A}_n(A)_{[0,i-1]}$ is a Toeplitz symmetric matrix with entries of size at most S. Solving such structured linear systems can be done in $\widetilde{O}(i)$ operations in $\mathbb{Q}$ [8, 9]. To handle the size of denominators, we take into account a factor iS essentially coming from the determinant of the system to solve, which makes this step in $\widetilde{O}(i^2S)$. Note that this does not make use of the fact that the system to solve is almost determined by the previous one. Section 7.2 exploits this additional structure to further improve the practical efficiency. Computing the roots of the quadratic equation amounts to a square root of a number of size $\widetilde{O}(i \cdot S)$, which is doable in $\widetilde{O}(i \cdot S)$ as well. Consequently a step is done in $\widetilde{O}(i^2S)$, so that the algorithm runs in $\widetilde{O}(n^3S)$.

A.2 Complexity of TowerRecovery$_{\mathcal{F}}$

We describe here the "integral" version of Algorithm 5.2. The idea is to handle the denominators by "keeping track" of them throughout the recursive execution of the algorithm, that is, passing them as extra input to the algorithm. At the top level, the starting denominator is 1, as should be for an input $u \in \mathcal{R}_n$. In particular, this makes the algorithm work in full generality for any algebraic number: give $u \in \mathcal{K}_n$, one may indeed always find d such that $U = du \in \mathcal{R}_n$, and run the algorithm for U. Below, recall that the trace and norm map are relative to the extension $\mathcal{K}_n|\mathcal{K}_{n/2}$.

At Step 4, if the algorithm followed the rational version, it would ouput $u^+ = \mathrm{Tr}(u)$ (up to conjugation). In the "integral" formulation, d is such that $du = U \in \mathcal{R}_n$, so Step 4 outputs $U^+ = du^+ = d\,\mathrm{Tr}(u) = \mathrm{Tr}(U)$. Similarly, and still up to conjugation, "rational" Step 5 outputs $\widetilde{u} = \frac{2\,\mathrm{N}(u)}{\mathrm{Tr}(u)}$, so the current version outputs $\widetilde{U} = m_{n/2-1}\widetilde{u} = \frac{m_{n/2-1}}{d}\frac{2\,\mathrm{N}(U)}{\mathrm{Tr}(U)}$. This shows that all the involved algebraic numbers are $\mathrm{Tr}(U)$ and $\mathrm{N}(U)$, both in $\mathcal{R}_{n/2}$. From the description of the algorithm, it is also seen that all the input of the recursive calls to TowerRecovery$_{\mathcal{F}}$ are subsets of the initial input, which shows that the successive inputs stay bounded at worst by $\log|m_{n-1}| \leq n\log\|U\|_\infty$.

We also observe that when the algorithm has "gone down" to $n = 2$, it returns immediately its first input. In particular, this shows that the cost of the algorithm is dominated by that of the

Algorithm A.1 $\mathsf{TowerRecovery}_{\mathcal{F}}(m_0, \ldots, m_{n-1}; d)$

Input: $m_0, \ldots, m_{n-1}, d \in \mathbb{Z}_+$ and $d = 1$ by default.
Output: $U \in \mathcal{R}_n$ such that U is totally positive and the principal minors of $\mathcal{F}_n(U)$ are $d^i m_i$'s ($0 \le i < n$).

1: **if** $n = 2$ **then**
2: **return** m_0.
3: **end if**
4: $U^+ \leftarrow \mathsf{TowerRecovery}_{\mathcal{F}}(m_0, \ldots, m_{\frac{n}{2}-1}; d)$ $\{U^+$ is a conjugate of $\mathrm{Tr}(U)/2\}$
5: $\widetilde{U} \leftarrow \mathsf{TowerRecovery}_{\mathcal{F}}(m_{\frac{n}{2}}, \ldots, m_{n-1}; m_{\frac{n}{2}-1})$ $\{\widetilde{U}$ is a conjugate of $\frac{m_{n/2-1}}{d} \frac{\mathrm{N}(U)}{\frac{1}{2}\mathrm{Tr}(U)}\}$
6: $\widetilde{U} \leftarrow \widetilde{U} \cdot \frac{d}{m_{\frac{n}{2}-1}}$
7: **for** $i = 1$ to $\frac{n}{2}$ **do**
8: $V \leftarrow U^+ \cdot (U^+ - \sigma_i(\widetilde{U}))$ $\{\sigma_i$ is the field automorphism $\zeta_{n/2} \mapsto \zeta_{n/2}^{2i-1}\}$
9: **if** $V \in \mathcal{R}_{n/2}$ **then**
10: $U^- \leftarrow \mathsf{TowerRoot}(\zeta_{n/2}^{-1} v)$ $\{U^-$ is a conjugate of $\pm\,\mathrm{Tr}(\zeta_n^{-1} U)\}$
11: **return** $U^+ + \zeta_n U^-$
12: **end if**
13: **end for**

square roots it has to compute, and of finding at each node of the tree a correct pair of conjugates. We first discuss the loop of Step 7. On the one hand, if U^+ and $\widetilde{U}$ corresponds to the same automorphism, then we must have $U^+\widetilde{U} = \mathrm{N}(U) \in \mathcal{R}_{n/2}$. This integrality condition transfers to the integrality of $V = (\frac{1}{2}\mathrm{Tr}(U))^2 - \mathrm{N}(U)$. On the other hand, with overwhelming probability, two non corresponding conjugates would give a number in $\mathcal{K}_{n/2}$. Therefore we can detect a correct pair by doing $\frac{n}{2}$ multiplications in $\mathcal{K}_{n/2}$.

Observe that Step 8 and Step 10 have input with size both being bounded by $\log \|\mathrm{N}(U)\|_\infty$. In Supplementary Material A.4, we show that $\mathsf{TowerRoot}$ has complexity $\mathsf{SQRT}(n, S) := O(n^2 S)$ for an input $s \in \mathcal{R}_n$ of size S. As multiplication between numbers of size S in $\mathcal{K}_{n/2}$ can be done by FFT in $\widetilde{O}(nS)$, and because at each level of the tree we do at most $\frac{n}{2}$ multiplications, we can assume the overall cost of the algorithm is given by the cost of computing all the square roots.

We have $\log \|\mathrm{N}(U)\|_\infty \le 2\log\|U\|_\infty + \log n$. At the i^{th} level of the tree, d ranges over the set $\{m_{(2k+1)n/2^i} : 0 \le k < 2^{i-1}\}$. We also have $\log |m_{(2k+1)n/2^i}| \le \frac{(2k+1)n}{2^i} \log\|U\|_\infty$, using for example Hadamard's inequality. Hence, computing the square roots at this level costs (up to some additive $\log n$ terms absorbed in the soft-O's):

$$\sum_{k=0}^{2^{i-1}-1} \mathsf{SQRT}\left(\frac{n}{2^i}, \frac{(2k+1)n}{2^{i-1}} \log\|U\|_\infty\right) = \widetilde{O}\left(\frac{n^3}{2^{3i}} \log\|U\|_\infty \sum_{k=0}^{2^{i-1}-1} (2k+1)\right)$$
$$= \widetilde{O}\left(\frac{n^3}{2^i} \log\|U\|_\infty\right).$$

Summing the cost of all levels gives a final complexity of $\widetilde{O}(n^3 \log\|U\|_\infty)$.

A.3 Complexity of CheckSqr

Recall that after computing the square root of $\mathrm{N}(s)$, we know $(-1)^b \mathrm{N}(t)$ for some $b \in \{0, 1\}$. From Equations (4), we see that $T_b := \mathrm{Tr}(s) + 2\cdot(-1)^b \mathrm{N}(t)$ is a square in $\mathcal{K}_{n/2}$ if and only if $b = 0$, in which case $T_b = \mathrm{Tr}(t)^2$. (Else, $\zeta_n^{-2} T_b$ is the square $\mathrm{Tr}(\zeta_n^{-1} t)^2$ in $\mathcal{K}_{n/2}$.) Hence we can recover b by checking whether T_b is a square in $\mathcal{K}_{n/2}$ or not. Next, observe that $\mathrm{N}(\zeta_n^2) = -\zeta_n^4 \in \mathcal{K}_{n/4}$, so

that either $\mathrm{N}(T_b)$ or $\zeta_n^{-4}\,\mathrm{N}(T_b)$ is a square in $\mathcal{K}_{n/4}$. In other words, we can distinguish the value of b by going down in the tower of field using relative norms, and lastly checking for squareness in $\mathbb{Q}(i)$.

Let $u = a + ib = (r + is)^2$ in $\mathbb{Z}[i]$. Then $b = 2rs$, so checking the parity of b gives a first immediate test for u to be a square. We also have $a = r^2 - s^2$, so that $r^4 - ar^2 - b^2/4 = 0$. Equivalently, if u is a square, then the integer polynomial $P = X^2 - aX - b^2/4$ must have two distinct integer roots r_1, r_2, one of which being r^2. Combined with $b^2/4 = (rs)^2 = r_1 r_2$, we see that the other root is s^2. We readily see that if $a \geq 0$ if and only if $|r| \geq |s|$ and that $b \neq 0$ if and only if r and s have opposite signs. We can then obtain a correct pair (r, s) without recomputing the corresponding square. Overall, whenever the discriminant $\mathrm{N}(u)$ of P is an integer square, then u will also be a square. This can be checked by computing $\sqrt{\mathrm{N}(u)}$ and checking if it is an integer, and can be done in time $\widetilde{O}(\max(\log|a|, \log|b|))$.

A.4 Complexity of TowerRoot

As we are not interested in logarithmic factors, and as the recursion depths is $\log n$, we will rely instead of Equation (5) on

$$\mathsf{SQRT}(n, S) = \mathsf{SQRT}\left(\frac{n}{2}, 2S\right) + 2\mathsf{SQRT}\left(\frac{n}{2}, S\right) + \widetilde{O}(nS).$$

Write $n = 2^\lambda$, and recall that $S = \log\|s\|_\infty$ where s is the input. We claim that for all $\ell \in [\lambda]$, we have

$$\mathsf{SQRT}(n, S) = \sum_{j=0}^{\ell} \binom{\ell}{j} 2^j \cdot \mathsf{SQRT}\left(\frac{n}{2^\ell}, 2^{\ell-j}S\right) + (2^\ell - 1)\widetilde{O}(nS).$$

From the description Equation (5), the formula holds for $\ell = 1$. We now proceed by induction, and assume the result is true for ℓ. Using that $2^\ell = \sum_{j=0}^{\ell}\binom{\ell}{j}$, we can rewrite

$$\mathsf{SQRT}(n, S) = \sum_{j=0}^{\ell} \binom{\ell}{j} \left(2^j \cdot \mathsf{SQRT}\left(\frac{n}{2^\ell}, 2^{\ell-j}S\right) + \widetilde{O}(nS)\right) - \widetilde{O}(nS). \tag{6}$$

Combined with the recurrence formula, we then have

$$\begin{aligned} 2^j \cdot \mathsf{SQRT}\left(\frac{n}{2^\ell}, 2^{\ell-j}S\right) &= 2^j \cdot \mathsf{SQRT}\left(\frac{n}{2^{\ell+1}}, 2^{\ell-j+1}S\right) \\ &\quad + 2^{j+1} \cdot \mathsf{SQRT}\left(\frac{n}{2^{\ell+1}}, 2^{\ell-j}S\right) \\ &\quad + \widetilde{O}(nS). \end{aligned}$$

Substituting in Equation (6), we obtain:

$$\begin{aligned}\mathsf{SQRT}(n,S) &= \sum_{j=0}^{\ell}\binom{\ell}{j}2^j\cdot\mathsf{SQRT}(\tfrac{n}{2^{\ell+1}},2^{\ell-j+1}S)\\ &\quad+\sum_{j=0}^{\ell}\binom{\ell}{j}2^{j+1}\cdot\mathsf{SQRT}(\tfrac{n}{2^{\ell+1}},2^{\ell-j}S)+\left(2\sum_{j=0}^{\ell}\binom{\ell}{j}-1\right)\widetilde{O}(nS)\\ &= \sum_{j=0}^{\ell+1}\left(\binom{\ell}{j}+\binom{\ell}{j-1}\right)2^j\cdot\mathsf{SQRT}(\tfrac{n}{2^{\ell}},2^{\ell-j+1}S)+(2^{\ell+1}-1)\widetilde{O}(nS),\end{aligned}$$

which gives the claim by standard binomial identities. Setting $\ell = \lambda$, and using the fact that computing integer square roots takes quasi-linear time in the size of the input, we now find

$$\begin{aligned}\mathsf{SQRT}(n,S) &= \widetilde{O}(n^2S)+\sum_{j=0}^{\lambda}\binom{\lambda}{j}\cdot 2^j\cdot\mathsf{SQRT}\big(1,2^{\lambda-j}(S+\lambda^2)\big)\\ &= \widetilde{O}(n^2S)+\widetilde{O}(nS)\sum_{j=0}^{\lambda}\binom{\lambda}{j},\end{aligned}$$

which shows the quasi-quadratic complexity of our algorithm.

B About the coefficients of $f\overline{f}$

The constant coefficient is $\|f\|^2$. As $f \hookleftarrow D_{\mathbb{Z}^n,r}$ with $r = 1.17\sqrt{\frac{q}{2n}}$, by Lemma 1.5 in [3], we have $\|f\|^2 > q$ with probability $\leq \left(\frac{\sqrt{2e}}{1.17}\exp(-\frac{1}{1.17^2})\right)^n \leq 0.96^n$.[7] For large n, we know with high probability that $\|f\|^2 \in (0,q)$, which allows to distinguish $\|f\|^2 \bmod q$ and $q + (\|f\|^2 \bmod q)$.

The non-constant coefficients are $\langle f, x^k f\rangle$ with $k > 0$. A routine computation shows that $\langle f, x^k f\rangle = \sum_{i\in\mathbb{Z}_n}\varepsilon_i f_i f_{i+k}$ where $\varepsilon_i \in \{\pm 1\}$. A inconvenience here is the dependency between the product variables involved in the sum. To avoid it, we separate the indices in two disjoint partitions $A_n(k), B_n(k)$ of $\mathbb{Z}_n$, which are exchanged by translation by k. Lemma 4 below shows that it is always possible. Hence, we can write

$$\langle f, x^k f\rangle = \sum_{i\in A_n(k)}\varepsilon_i f_i f_{i+k} + \sum_{i\in B_n(k)}\varepsilon_i f_i f_{i+k} = \langle \mathbf{f}_A, \mathbf{f}'_B\rangle + \langle \mathbf{f}'_A, \mathbf{f}_B\rangle$$

where $(A_n(k), B_n(k))$ are as described above, and $\mathbf{f}_A = (\varepsilon_i f_i)_{i\in A_n(k)}$, $\mathbf{f}'_A = (f_{i+k})_{i\in B_n(k)}$. Similarly, we use the notation $\mathbf{f}_B, \mathbf{f}'_B$. Since all f_i's are independently sampled from $D_{\mathbb{Z},r}$ with $r = 1.17\sqrt{\frac{q}{2n}}$, then $\mathbf{f}_A$ (resp. $\mathbf{f}_B$) and $\mathbf{f}'_B$ (resp. $\mathbf{f}'_A$) are independent. By [33, Lemma 4.3] we have that $|\langle \mathbf{f}_A, \mathbf{f}'_B\rangle| \leq r\cdot t\sqrt{2\pi}\|\mathbf{f}_B\|, |\langle \mathbf{f}'_A, \mathbf{f}_B\rangle| \leq r\cdot t\sqrt{2\pi}\|\mathbf{f}_A\|$ with probability $\geq p(t) =$

[7] We let $L = (r\sqrt{2\pi})^{-1}\cdot\mathbb{Z}$ and $c = (1.17\sqrt{\pi})^{-1}$ in the Lemma 1.5 of [3].

$1 - 2\exp(-\pi t^2)$.[8] By the fact that $\|\mathbf{f}_A\|^2 + \|\mathbf{f}_B\|^2 = \|f\|^2 \leq 1.17^2 q$, we have

$$\begin{aligned}
|\langle f, x^k f\rangle| &\leq |\langle \mathbf{f}_A, \mathbf{f}'_B\rangle| + |\langle \mathbf{f}'_A, \mathbf{f}_B\rangle| \\
&\leq r \cdot t\sqrt{2\pi}(\|\mathbf{f}_A\| + \|\mathbf{f}_B\|) \\
&\leq 2r \cdot t\sqrt{\pi}\|f\| \\
&\leq q \cdot 1.37 \cdot t\sqrt{2\pi/n}
\end{aligned}$$

with probability $\geq p(t)^2$. Choosing $t = \sqrt{\frac{n}{50}}$ ensures that all coefficients except the constant one of $f\overline{f}$ are in $(-q/2, q/2)$ with high probability.

Lemma 4. *Let $n = 2^l$ with $l \geq 1 \in \mathbb{Z}$. For any $k \in \mathbb{Z}_n \backslash \{0\}$, there are two sets $A_n(k), B_n(k) \subseteq \mathbb{Z}_n$ such that:*

1. *$A_n(k) \bigcap B_n(k) = \emptyset$ and $\sharp(A_n(k)) = \sharp(B_n(k)) = n/2$;*
2. *if $x \in A_n(k)$, then $x + k \in B_n(k)$; if $x \in B_n(k)$, then $x + k \in A_n(k)$.*

Proof. We prove by induction. For $n = 2$, it is clear that $(A_n(1), B_n(1)) = (\{0\}, \{1\})$ satisfies all conditions for the only $k = 1$. Suppose that lemma holds for n', now we proceed to the case of $n = 2n'$.

Case 1: For $k = 1 \bmod 2$, it is easy to see that $A_n(k) = \{i \in \mathbb{Z}_n \mid i = 0 \bmod 2\}$ and $B_n(k) = \{i \in \mathbb{Z}_n \mid i = 1 \bmod 2\}$ satisfy all conditions.

Case 2: For $k = 2k'$ with $k' \in \mathbb{Z}_{n'}$, we let

$$A_n(k) = \{2i \mid i \in A_{n'}(k')\} \cup \{2i + 1 \mid i \in A_{n'}(k')\},$$

$$B_n(k) = \{2i \mid i \in B_{n'}(k')\} \cup \{2i + 1 \mid i \in B_{n'}(k')\}.$$

Clearly, $\sharp(A_n(k)) = \sharp(B_n(k)) = n/2$. For $2i \in A_n(k)$, we know $i \in A_{n'}(k')$ and then $i + k' \in B_{n'}(k')$. Thus $(2i + k) = 2(i + k') \in B_n(k)$. For $2i + 1 \in A_n(k)$, we can prove $2i + 1 + k \in B_n(k)$ similarly. The same argument hold for $B_n(k)$ and thus $(A_n(k), B_n(k))$ satisfies the second condition. We complete the proof. □

[8] The factor $\sqrt{2\pi}$ comes from the scaling factor between Gaussian width and standard deviation.