Cryptanalysis of the PEREGRINE Lattice-Based Signature Scheme

Xiuhan Lin[1], Moeto Suzuki[2], Shiduo Zhang[3], Thomas Espitau[4], Yang Yu[3,5,6,7], Mehdi Tibouchi[2,8], and Masayuki Abe[2,8]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
xhlin@mail.sdu.edu.cn
[2] Kyoto University, Kyoto, Japan
suzuki.moeto.56f@st.kyoto-u.ac.jp
[3] Institute for Advanced Study, Tsinghua University, Beijing, China
zsd19@mails.tsinghua.edu.cn
[4] PQShield SAS, Paris, France
t.espitau@gmail.com
[5] BNRist, Tsinghua University, Beijing, China
yu-yang@mail.tsinghua.edu.cn
[6] Zhongguancun Laboratory, Beijing, China
[7] National Financial Cryptography Research Center, Beijing, China
[8] NTT Social Informatics Laboratories, Tokyo, Japan
{mehdi.tibouchi,msyk.abe}@ntt.com

Abstract. The PEREGRINE signature scheme is one of the candidates in the ongoing Korean post-quantum cryptography competition. It is proposed as a high-speed variant of FALCON, which is a hash-and-sign signature scheme over NTRU lattices and one of the schemes selected by NIST for standardization. To this end, PEREGRINE replaces the lattice Gaussian sampler in the FALCON signing procedure with a new sampler based on the *centered binomial* distribution. While this modification offers significant advantages in terms of efficiency and implementation, it does not come with a provable guarantee that signatures do not leak information about the signing key. Unfortunately, lattice-based signature schemes in the hash-and-sign paradigm that lack such a guarantee (such as GGH, NTRUSign or DRS) have generally proved insecure.

In this paper, we show that PEREGRINE is no exception, by demonstrating a practical key recovery attack against it. We observe that the distribution of PEREGRINE signatures is a *hidden transformation* of some public distribution and still leaks information about the signing key. By adapting the parallelepiped-learning technique of Nguyen and Regev (Eurocrypt 2006), we show that the signing key can be recovered from a relatively small number of signatures. The learning technique alone yields an approximate version of the key, from which we can recover the exact key using a decoding technique due to Thomas Prest (PKC 2023).

For the reference implementation (resp. the official specification version) of PEREGRINE–512, we fully recover the secret key with good probability in a few hours given around 25,000 (resp. 11 million) signature samples.

Keywords: Cryptanalysis · Lattice-based signature · Statistical learning · NTRU

1 Introduction

As the potential advent of large-scale quantum computers puts all currently deployed public-key cryptography at risk, preparing the transition to post-quantum cryptography (PQC) is of great importance. To that end, NIST initiated its PQC standardization process in 2016, and announced in 2022 an initial batch of algorithms to be standardized by 2024, while other schemes remain under consideration for future standardization. Concurrently, other standardization bodies have launched their own processes and competitions for post-quantum cryptography, such as those in China[9], recently concluded, and the Republic of Korea[10], currently ongoing. In all these standardization efforts, constructions based on structured lattices have been

[9] https://www.cacrnet.org.cn/site/content/854.html
[10] https://kpqc.or.kr/

of particular interest, as their good balance of performance and bandwidth requirements have made them some of the top contenders both for public-key encryption/KEMs and signatures.

Lattice-based signature candidates can be roughly divided into two families according to their design paradigm: they either rely on the Fiat–Shamir heuristic (often but not always paired with Lyubashevsky's aborting technique) or on the hash-and-sign framework based on lattice trapdoors. The two lattice-based signatures already selected for standardization in the NIST process, namely Dilithium [LDK+22] and Falcon [PFH+22], each represent one of those two paradigms: Dilithium is a Fiat–Shamir with aborts scheme, while Falcon is a hash-and-sign construction. They both have their pros and cons: Dilithium is a simpler design that is easier to implement securely, whereas Falcon tends to be faster and more compact at the cost of a much greater complexity of implementation and a somewhat reduced versatility in terms of parameter selection.

Hash-and-sign lattice-based signatures in general rely on the construction of a certain lattice with a "good" and a "bad" basis, that play the role of the secret signing key and the public verification key respectively. The good basis, also called the *trapdoor*, has relatively short and fairly orthogonal vectors, and as such can be used to approximate the closest vector problem to a good approximation factor. The bad basis on the other hand consists of much larger vectors, and is of little help in finding close vectors, even though it can still of course be used to check lattice membership. The signing procedure can then be described roughly as follows. The message to be signed is hashed to a random point in the ambient space of the lattice, and the signer uses the trapdoor to compute some lattice point close to that random point, and outputs it as the signature. The verification consists in checking that the signature is indeed a lattice point (which can be done with the "bad" public basis) and that it is sufficiently close to the hashed message.

That general framework is the one followed by the first proposed lattice-based signature schemes in the late 1990s, like GGH [GGH97] and NTRUSign [HHP+03]. Unfortunately, those early constructions turned out to be insecure: Nguyen and Regev [NR06] showed that each signature would leak some information about the secret trapdoor, and that a few tens of thousands of signatures could suffice to reconstruct the entire signing key. Several heuristic countermeasures against this kind of statistical attacks were subsequently proposed and promptly broken again, until Gentry, Peikert, and Vaikuntanathan [GPV08] finally presented a provably secure solution to that issue. The crux of what is now called the GPV framework is to randomize the signature generation in such a way that signatures follow a distribution *independent* of the trapdoor (usually a certain discrete Gaussian distribution supported on the lattice). This is the approach followed by most of the secure hash-and-sign lattice-based signatures proposed afterward, including Falcon. The reliance on Gaussian sampling, however, is also one of the source of Falcon's greater complexity compared to schemes like Dilithium.

Several variants of Falcon [EFG+22, KTW+22, ENS+23] have recently been proposed to mitigate its shortcomings with respect to implementation complexity while remaining firmly within the GPV framework and its provable security guarantees. Nevertheless, trying to do away with the GPV framework altogether and adopt more efficient countermeasures against statistical attacks may appear like a tempting choice. This idea is the basic design principle of the PEREGRINE signature scheme, one of the candidates in the ongoing first round of the Korean PQC competition. PEREGRINE avoids the Gaussian sampling in its signature generation procedure entirely. Instead, it uses Babai's round-off algorithm [Bab86] to first compute a vector $\mathbf{v}$ close to the hashed message and then adds a random noise $\mathbf{e} = \mathbf{B}\mathbf{r}$ to get the signature $\mathbf{s} = \mathbf{v} + \mathbf{e}$ where $\mathbf{B}$ is the trapdoor and $\mathbf{r}$ is a random vector with coefficients following a *centered binomial* distribution. The addition of $\mathbf{e}$ is presented as an effective countermeasure against the Nguyen–Regev statistical attack [NR06] mentioned above. As noted in [SKLN22], the rationale underlying this choice is that the centered binomial distribution has "characteristics similar to a discrete Gaussian", although it does not offer a proof of security.

Hash-and-sign lattice signatures lacking such a proof, including the GGH and NTRUSign schemes already mentioned, as well as newer proposals like DRS [PSDS17], have generally been broken by statistical attacks [GS02, NR06, DN12, YD18]. Since PEREGRINE employs an additional countermeasure to specifically thwart this class of attacks, it is natural to analyze to what extent it succeeds in doing so.

Our contributions. In this work, we show that, unfortunately, PEREGRINE is no exception and can indeed be defeated with a statistical learning attack similar to [NR06]. As experimental validation, we implement a practical key recovery attack that effectively recovers the PEREGRINE signing key from a few tens of thousands to a few million signature samples, depending on the precise variant under consideration.

As mentioned before, the PEREGRINE signature can be written as $\mathbf{s} = \mathbf{v} + \mathbf{B}\mathbf{r}$ where $\mathbf{v}$ is in the parallelepiped spanned by the trapdoor $\mathbf{B}$, and $\mathbf{r}$ is drawn from a centered binomial distribution. The support of

PEREGRINE signatures is now a set of *adjacent parallelepipeds* rather than a sole parallelepiped, and these adjacent parallelepipeds are labeled by $\mathbf{r}$. For random messages, the signatures are uniformly distributed in each parallelepiped and the probability of a random signature in a certain parallelepiped only depends on the label $\mathbf{r}$. We observe that the signature distribution can be obtained by applying the linear transformation of $\mathbf{B}$ on some *publicly known* distribution. We call this general problem the *hidden transformation* problem (HTP), and propose to revisit the classical parallelepiped-learning technique of [NR06] in that context.

We show that this attack applies to PEREGRINE, allowing to fully exploit the statistical leak on $\mathbf{B}$, and demonstrate that it leads to a practical key recovery. In particular, we notice that the reference implementation of PEREGRINE does not coincide with the algorithmic description in the official specification. Hence, we mount the attack on those two versions of PEREGRINE, and manage to break both of them. Our attack combines several techniques in the literature. We make use of the gradient descent algorithm in [TW20] as an improvement of the original method in [NR06]. We also employ a decoding technique inspired by Prest [Pre23] to recover the exact key from the approximation derived from statistical learning. Finally, we show that around 25,000 (resp. 11 million) signatures suffice to successfully recover the full key within a few hours with good probability for the reference implementation (resp. the official specification version) of PEREGRINE–512.

Roadmap. We start in Section 2 with the notations and background. Section 3 briefly describes the PEREGRINE signature scheme. In Section 4, we extend the parallelepiped-learning technique to "learning a hidden transformation". In Section 5, we analyze and carry out practical attacks against both the reference implementation and the official specification of PEREGRINE–512. Finally, we conclude in Section 6 and suggest some possible avenues for future work.

2 Preliminaries

2.1 Notation

We denote by $\mathrm{GL}_n(\mathbb{R})$ be the group of $n \times n$ invertible matrices with real coefficients and $\mathbb{S}^{n-1}$ be the unit sphere of $\mathbb{R}^n$.

We describe (column) vectors in bold lowercase and write b_i as the i-th coordinate of vector $\mathbf{b}$, i.e. $\mathbf{b} = (b_1, \ldots, b_n)$. Given $\mathbf{a} = (a_1, \ldots, a_n)$, $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$, the inner product is $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i b_i$. For $\mathbf{a} \in \mathbb{R}^n$, the ℓ_2-norm is $\|\mathbf{a}\| = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle}$, the ℓ_1-norm is $\|\mathbf{a}\|_1 = \sum_i |a_i|$ and ℓ_∞-norm is $\|\mathbf{a}\|_\infty = \max_i |a_i|$. The operation $\lfloor x \rceil$ means rounding x to the closest integer and is naturally extended to the vector $\mathbf{x}$ by taking rounding coefficient-wisely. We describe matrices in bold uppercase and denote by $\mathbf{b}_i$ for the i-th column of matrix $\mathbf{B}$, i.e. $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. We use $\mathbf{B}^t$ (resp. $\mathbf{B}^{-1}$) to denote the transpose (resp. inverse) of $\mathbf{B}$. Let $\mathbf{I}_n$ be $n \times n$ identity matrix. Given a differentiable function $f \colon \mathbb{R}^n \to \mathbb{R}$, the gradient of f at $\mathbf{w} \in \mathbb{R}^n$ is $\nabla f(\mathbf{w}) = (\frac{\partial f}{\partial x_1}(\mathbf{w}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{w}))$.

2.2 Lattices

A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$. It is the set of all integer combinations of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$, i.e. $\mathcal{L} = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$. The matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called the basis and n the rank of $\mathcal{L}$. When $n = m$, the lattice is said to be full-rank. We write $\mathcal{L}(\mathbf{B})$ to denote the lattice generated by a basis $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$. Given $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$, the parallelepiped spanned by $\mathbf{B}$ is $\mathcal{P}(\mathbf{B}) = \left\{\mathbf{x}\mathbf{B} \mid \mathbf{x} \in \left[-\frac{1}{2}, \frac{1}{2}\right)^n\right\}$. The parallelepiped $\mathcal{P}(\mathbf{B})$ is a fundamental region of the lattice $\mathcal{L}(\mathbf{B})$.

2.3 Statistics and Probability

For a distribution D, we write $y \leftarrow D$ when the random variable y is sampled from D. We also write $y \sim D$ a random variable y following the distribution D. Let $U(S)$ be the uniform distribution over the set S. Let $\#S$ be the number of elements in the set S. We denote by $\mathbb{E}[y]$ the expectation of the random variable y. A distribution D over $\mathbb{R}$ is called *centered* when $\mathbb{E}_{y \leftarrow D}[y] = 0$. For a distribution D over $\mathbb{R}^n$, we denote by $\mathbf{Cov}[D] = \mathbb{E}_{\mathbf{x} \leftarrow D}[\mathbf{x}^t \mathbf{x}]$ its covariance matrix.

Given an even integer μ, the *centered binomial* distribution B_μ is defined over $[-\frac{\mu}{2}, \frac{\mu}{2}] \cap \mathbb{Z}$ by the probability density function $P[X = x] = \frac{\mu!}{(\frac{\mu}{2}+x)!(\frac{\mu}{2}-x)!} \cdot 2^{-\mu}$.

Table 1. Parameters of PEREGRINE.

Symbol	Description
$n/2$ (a power of two)	degree of the underlying ring $\mathcal{R} = \mathbb{Z}[x]/(x^{n/2}+1)$
q	modulus
(μ_1, μ_2)	parameters of the binomial distributions in signing
$\lfloor \beta^2 \rfloor$	signature acceptance bound

Algorithm 1: Sign

Input: A message msg, an NTRU trapdoor $\mathbf{B}_{f,g}$
Output: A signature (r, s_2)

$r \xleftarrow{\$} \{0,1\}^{320}$, $c \leftarrow \mathsf{H}(msg \| r)$
$(I_1, I_2) \leftarrow \left(\left\lfloor -\frac{cF}{q} \right\rceil, \left\lfloor \frac{cf}{q} \right\rceil \right)$
repeat
  $(J_{1,0}, \ldots, J_{1,n/2-1}) \leftarrow B_{\mu_1}^{n/2}, (J_{2,0}, \ldots, J_{2,n/2-1}) \leftarrow B_{\mu_2}^{n/2}$
  $J_1 \leftarrow \sum_{i=0}^{n/2-1} J_{1,i} \cdot x^i$, $J_2 \leftarrow \sum_{i=0}^{n/2-1} J_{2,i} \cdot x^i$
  $\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \leftarrow \begin{pmatrix} c \\ 0 \end{pmatrix} - \mathbf{B}_{f,g} \cdot \begin{pmatrix} I_1 + J_1 \\ I_2 + J_2 \end{pmatrix}$
until $\|(s_1, s_2)\| \leq \lfloor \beta^2 \rfloor$
return (r, s_2)

2.4 Cyclotomic rings and NTRU

Let $\mathcal{R} = \mathbb{Z}[x]/(x^{n/2}+1)$ with $n \geq 4$ a power of 2. Given $h \in \mathcal{R}$ and q a rational prime with h invertible modulo q, the lattice $\mathcal{L}_{NTRU} = \{(s_1, s_2) \in \mathcal{R}^2 \mid s_1 + s_2 h = 0 \bmod q\}$ is called an NTRU lattice. In a typical NTRU cryptosystem, the public key is $h = g/f \bmod q$ where (f, g) is a pair of short polynomials in $\mathcal{R}$ and used as the secret key. For short $(F, G) \in \mathcal{R}^2$ such that $fG - gF = q$, $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ -f & -F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ is an NTRU trapdoor basis of $\mathcal{L}_{NTRU}$.

3 The PEREGRINE Signature Scheme

Let us briefly describe the PEREGRINE signature scheme. We omit some details that are not necessary for understanding our work and refer to [SKLN22] for the complete description. PEREGRINE is specified by the parameters of Table 1.

As an NTRU-based hash-and-sign signature scheme, PEREGRINE uses an NTRU trapdoor basis $\mathbf{B}_{f,g}$ as the secret key and the public key is $h = g/f \bmod q$.

Signing procedure. The signing algorithm of PEREGRINE is described in Algorithm 1. It proceeds in two steps. The first step is in essence Babai's round-off algorithm [Bab86] outputting some integer vector (I_1, I_2) such that $\mathbf{B}_{f,g} \cdot \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$ is close to the hashed message $\begin{pmatrix} c \\ 0 \end{pmatrix}$. This is implemented with a so-called ModDown function, and the details are irrelevant to our attack. The second step, perhaps inspired by the Gaussian sampling techniques [GPV08, Pei10], is randomizing the close vector $\mathbf{B}_{f,g} \cdot \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$, which is attempted to defeat the parallelepiped-learning attack [NR06]. The randomization is performed by adding a binomial vector (J_1, J_2) to (I_1, I_2) without using Gaussian distribution.

Concrete parameters. This work focuses on the parameter set of PEREGRINE–512 that was claimed to reach NIST security level I, in which $(n/2, q) = (512, 12289)$. We noted some discrepancies between the reference *implementation* and the official *specification*.

1. As per the specification, the key generation of PEREGRINE is almost the same with that of Falcon, except that the coefficients of (f, g) are drawn from the binomial distribution B_{26}. The key generation should check if the Gram–Schmidt norms of $\mathbf{B}_{f,g}$ are bounded by $1.17\sqrt{q}$ to ensure the optimal trapdoor quality [DLP14]. However, this check is commented out in the reference implementation, which yields the risk of weak keys. We follow the key generation with such a check, while it would not affect our attack.
2. While the specification suggests $\mu_1 = \mu_2 = 26$, the reference implementation in effect works with $(\mu_1, \mu_2) = (6, 0)$. We mount the attack on both the reference implementation and the specification version. The reference implementation uses much smaller (μ_1, μ_2) narrowing the support of signatures, which greatly facilitates the attack in practice. However, larger binomial parameters in the specification still cannot be an effective countermeasure.

Signature distribution. The signing procedure of PEREGRINE boils down to solving the approximate CVP in the NTRU lattice. The signature[11] (s_1, s_2) is the difference between the CVP solution to the target $(c, 0)$. It can be rewritten as

$$\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \mathbf{B}_{f,g} \cdot \begin{pmatrix} R_1 - J_1 \\ R_2 - J_2 \end{pmatrix}$$

where $\begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \mathbf{B}_{f,g}^{-1} \cdot \begin{pmatrix} c \\ 0 \end{pmatrix} - \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$. It is known that (R_1, R_2) is uniformly distributed over $\left[-\frac{1}{2}, \frac{1}{2}\right)^n$ over the randomness of signed messages, then the distribution of (s_1, s_2) is a hidden linear transformation (i.e. $\mathbf{B}_{f,g}$) of a known distribution. As a consequence, the statistics of (s_1, s_2) would leak secret information, which opens up the avenue of cryptanalysis.

4 Learning a Hidden Transformation

As shown in Section 3, the distribution of PEREGRINE signatures is a hidden linear transformation of some known distribution. Learning the hidden transformation enables a key recovery attack against PEREGRINE.

In this section, we extend the Nguyen-Regev parallelepiped-learning attack [NR06] to more general *Hidden Transformation Problem* (HTP) defined as follows.

Definition 1 (HTP_D)**.** *Let D be a public distribution over $\mathbb{R}^n$. Given a hidden matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathrm{GL}_n(\mathbb{R})$ and a certain number of independent samples $\mathbf{y} = \mathbf{B}\mathbf{x}$ with $\mathbf{x} \leftarrow D$, find an approximation of $\pm\mathbf{b}_i$'s.*

4.1 The Algorithmic Framework

As our main use case is for D being the joint distribution of $(x_1, \ldots, x_n)$ with x_i independently drawn from some centered distribution D_i, we choose to present our exploitation in this restricted setting for the sake of clarity. The adaptation to a general form of distribution is straightforward from there.

4.1.1 Distribution deformation.

Let σ_i be the standard deviation of D_i and $D(\mathbf{B})$ denote the distribution of $\mathbf{y} = \mathbf{B}\mathbf{x}$ with $\mathbf{x} \leftarrow D$. We first describe the covariance of the transformed distribution $D(\mathbf{B})$.

Lemma 1. *Let $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$, then $\mathbf{Cov}[D(\mathbf{B})] = \mathbf{B}\mathbf{\Sigma}\mathbf{B}^t$ where $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$.*

Proof. For $\mathbf{y} \sim D(\mathbf{B})$, let $\mathbf{y} = \mathbf{B}\mathbf{x}$ where $\mathbf{x} \sim D$. Then we have

$$\mathbf{Cov}[D(\mathbf{B})] = \mathbb{E}[\mathbf{y}\mathbf{y}^t] = \mathbb{E}[\mathbf{B}\mathbf{x}\mathbf{x}^t\mathbf{B}^t] = \mathbf{B}\,\mathbb{E}[\mathbf{x}\mathbf{x}^t]\mathbf{B}^t.$$

Since D_i's are centered, then $\mathbb{E}[\mathbf{x}\mathbf{x}^t] = \mathbf{\Sigma}$ and the proof is completed. □

The covariance leakage shown in Lemma 1 allows us to reduce the general hidden transformation problem to the case in which the covariance leakage is $\mathbf{I}_n$, by applying the exact matrix to unbias the distribution.

[11] The term s_1 can be recovered from the actual signature s_2 along with the NTRU public key and the hashed message.

Algorithm 2: Deform

Input: A distribution D and a set S of samples drawn from $D(\mathbf{B})$.
Output: A matrix $\mathbf{L}$ deforms samples from $D(\mathbf{B})$ into samples from a distribution close to $D(\mathbf{C})$ s.t. $\mathbf{C} = \mathbf{LB}$ and $\mathbf{Cov}[D(\mathbf{C})] = \mathbf{I}_n$.

1 Using the samples in S to compute an approximation $\mathbf{K}$ of $\mathbf{Cov}[D(\mathbf{B})]$
2 Compute $\mathbf{K}^{-1}$ and $\mathbf{P}$ such that $\mathbf{K}^{-1} = \mathbf{PP}^t$
3 **return** $\mathbf{L} = \mathbf{P}^t$

Lemma 2. *Let* $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$ *and* $\mathbf{K} = \mathbf{Cov}[D(\mathbf{B})]$. *Let* $\mathbf{P} \in \mathrm{GL}_n(\mathbb{R})$ *such that* $\mathbf{PP}^t = \mathbf{K}^{-1}$. *Then the distribution of* $\mathbf{P}^t\mathbf{y}$ *with* $\mathbf{y} \sim D(\mathbf{B})$ *is* $D(\mathbf{C})$ *with* $\mathbf{C} = \mathbf{P}^t\mathbf{B}$ *such that* $\mathbf{Cov}[D(\mathbf{C})] = \mathbf{I}_n$. *In particular,* $\mathbf{C}$ *is orthogonal when* $\mathbf{Cov}[D] = \mathbf{I}_n$.

Proof. Lemma 1 shows that $\mathbf{K} = \mathbf{B\Sigma B}^t$ and then $\mathbf{K}^{-1} = \mathbf{B}^{-t}\mathbf{\Sigma}^{-1}\mathbf{B}^{-1}$. By definition, $\mathbf{y} \sim D(\mathbf{B})$ can be written into $\mathbf{y} = \mathbf{Bx}$ with $\mathbf{x} \sim D$. It follows that $\mathbf{P}^t\mathbf{y} = \mathbf{P}^t\mathbf{Bx}$ follows the distribution $D(\mathbf{C})$ with $\mathbf{C} = \mathbf{P}^t\mathbf{B}$. Moreover, $\mathbf{Cov}[D(\mathbf{C})] = \mathbf{C\Sigma C}^t = \mathbf{P}^t\mathbf{KP} = \mathbf{I}_n$. □

Lemma 2 translates directly into Algorithm 2, when we don't have access to the covariance of $D(\mathbf{B})$ itself, but we only have a family of samples of this distribution. Hence, we first approximate the covariance as precisely as possible, and then compute the square root as in Lemma 1 and apply the transformation to correct it. The estimation of the covariance is dependent of the prior we have on D. In the most general context, we rely on the computation of the *sample covariance matrix*[12]:

$$\mathbf{K} = \frac{1}{\#S - 1} \sum_{\mathbf{x} \in S} \mathbf{x}\mathbf{x}^t.$$

4.1.2 Mounting the recovery. We next generalize the statistical analysis of the parallelepiped-learning attack [NR06] to the HTP_D setting. Algorithm 2 reduces the HTP instance regarding $(D, \mathbf{B})$ to the one regarding $(D, \mathbf{C})$ such that $\mathbf{Cov}[D(\mathbf{C})] = \mathbf{I}_n$. Let $\mathbf{D} = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ and D' be the distribution of $\mathbf{D}^{-1}\mathbf{x}$ for $\mathbf{x} \sim D$, then $\mathbf{Cov}[D'] = \mathbf{I}_n$. Let $\mathbf{C}' = \mathbf{CD}$, then $D(\mathbf{C}) = D'(\mathbf{C}')$ and $\mathbf{C}'$ is orthogonal. Hence, the rest of this section focuses on the HTP instance in which the hidden matrix $\mathbf{C}$ is orthogonal and the public distribution D satisfying $\mathbf{Cov}[D] = \mathbf{I}_n$.

Let $\mathbf{m} = \sum_{i=1}^n z_i\mathbf{c}_i$ with $z_i \sim D_i$, then $\mathbf{m} \sim D(\mathbf{C})$. The fourth moment (also called *kurtosis*) of $D(\mathbf{C})$ over $\mathbf{w} \in \mathbb{R}^n$ is defined as

$$M_{D(\mathbf{C}),4}(\mathbf{w}) = \mathbb{E}[\langle \mathbf{m}, \mathbf{w}\rangle^4].$$

Let $\alpha_i = \mathbb{E}[z_i^4]$. It holds that

$$\begin{aligned} M_{D(\mathbf{C}),4}(\mathbf{w}) = \mathbb{E}[\langle \mathbf{m}, \mathbf{w}\rangle^4] &= \mathbb{E}\left[\left(\sum_{i=1}^n z_i\langle \mathbf{c}_i, \mathbf{w}\rangle\right)^4\right] \\ &= \sum_{i=1}^n \mathbb{E}[z_i^4]\langle \mathbf{c}_i, \mathbf{w}\rangle^4 + 3\sum_{i\neq j}\langle \mathbf{c}_i, \mathbf{w}\rangle^2\langle \mathbf{c}_j, \mathbf{w}\rangle^2 \\ &= 3\|\mathbf{w}\|^4 - \sum_{i=1}^n (3 - \alpha_i)\langle \mathbf{c}_i, \mathbf{w}\rangle^4. \end{aligned}$$

Without loss of generality, we assume $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n$. Lemma 3 shows the local minima of the fourth moment over all unit $\mathbf{w}$'s.

Lemma 3. *Suppose that* $\alpha_i < 3$ *for all* $1 \leq i \leq n$, *the local minimum of* $M_{D(\mathbf{C}),4}(\mathbf{w})$ *over all unit vectors* $\mathbf{w}$ *is obtained at* $\pm\mathbf{c}_1, \ldots, \pm\mathbf{c}_n$. *There are no other local minima.*

[12] We recall that we supposed here that the distributions are centered. The term $\#S - 1$ is the standard *Bessel correction* and is necessary to get an unbiased estimator.

Proof. Since $\mathbf{C}$ is orthogonal, $M_{D(\mathbf{I}),4}(\mathbf{w}) = 3\|\mathbf{w}\|^4 - \sum_{i=1}^n (3-\alpha_i) w_i^4$ has the same local extrema with $M_{D(\mathbf{C}),4}(\mathbf{w})$. Under the orthogonal transformation, the local extreme points of two functions are one-to-one.

Let $\beta_i = 3 - \alpha_i > 0$, then the minima of $M_{D(\mathbf{I}),4}(\mathbf{w})$ are the maxima of $f(\mathbf{w}) = \sum_{i=1}^n \beta_i w_i^4$ as $\|\mathbf{w}\| = 1$. It suffices to study the local maxima of $f(\mathbf{w})$. Using the method of Lagrange multipliers, we consider

$$F(\mathbf{w}) = \sum_{i=1}^n \beta_i w_i^4 - \lambda \left(\sum_{i=1}^n w_i^2 - 1 \right).$$

In order to meet $\nabla F = \mathbf{0}$, each w_i is either zero or $\pm\sqrt{\frac{\lambda}{2\beta_i}}$ and λ is determined by the number of non-zero w_i's. These points can be divided into two sets: the points in the first set are $\pm\mathbf{e}_i$'s where $\mathbf{e}_i$ is the i-th column of the identity matrix and the other set $\mathcal{I}$ contains the points with at least two non-zero coefficients.

We first prove that $\pm\mathbf{e}_i$ is a local maximum. Let $\mathbf{w} = (1-\epsilon_i)\mathbf{e}_i + \sum_{j\neq i} \epsilon_j \mathbf{e}_j$ be some vector in $\mathcal{B} = \{\mathbf{w} : \|\mathbf{w} - \mathbf{e}_i\|^2 < \epsilon, \|\mathbf{w}\| = 1, \epsilon_i > 0\}$. Then

$$\begin{aligned}
f(\mathbf{w}) &= \beta_i (1-\epsilon_i)^4 + \sum_{j\neq i} \beta_j \epsilon_j^4 \\
&\leq \beta_i - \beta_i \epsilon_i + \max_{j\neq i} |\epsilon_j| \cdot \sum_{j\neq i} \beta_j \epsilon_j^2 \\
&\leq \beta_i - \beta_i \epsilon_i + \max_{j\neq i} |\epsilon_j| \cdot \beta_1 \left(1 - (1-\epsilon_i)^2\right) \\
&\leq \beta_i - \beta_i \epsilon_i + \max_{j\neq i} |\epsilon_j| \cdot 2\beta_1 \epsilon_i \\
&\leq \beta_i + \left(\max_{j\neq i} |\epsilon_j| \cdot 2\beta_1 - \beta_i \right) \epsilon_i \\
&< \beta_i = f(\mathbf{e}_i).
\end{aligned}$$

Above inequalities hold when $\epsilon < \min\{\frac{1}{4}, \frac{\beta_i}{2\beta_1}\}$, thus $\pm\mathbf{e}_i$ is indeed the local maximum of $f(\mathbf{w})$.

Then we prove that $\mathbf{w} = \sum_{i=1}^n \eta_i \mathbf{e}_i \in \mathcal{I}$ is not a local maximum. Let $\mathcal{K}_\mathbf{w} = \{\mathbf{e}_i : \langle \mathbf{w}, \mathbf{e}_i \rangle \neq 0\}$ and $k = \#\mathcal{K}_\mathbf{w}$. Let $\mathcal{S}$ be the subspace spanned by $\mathbf{e}_i \in \mathcal{K}_\mathbf{w}$ and $|_\mathcal{S}$ denote the directly dimensional reduction into $\mathcal{S}$. We next prove that $\mathbf{w}|_\mathcal{S}$ forms the local minima of $f|_\mathcal{S}$, which implies $\mathbf{w}$ is no longer the local maximum of f. The bordered Hessian matrix of Lagrange function $F|_\mathcal{S}$ on the point $\mathbf{w}|_\mathcal{S}$ has the form $\mathbf{H} = \begin{pmatrix} 0 & 2\mathbf{w}^t|_\mathcal{S} \\ 2\mathbf{w}|_\mathcal{S} & d\mathbf{I} \end{pmatrix} \in \mathbb{R}^{(k+1)\times(k+1)}$ and $d > 0$ is a constant related to η_i. Notice that $\det(\mathbf{H}) = \det(d\mathbf{I}) \det(0 - \mathbf{w}^t|_\mathcal{S} (d\mathbf{I})^{-1} \mathbf{w}|_\mathcal{S}) = -d^{k-1} < 0$. Similarly, all leading principle minors of $\mathbf{H}$ have a negative determinant. The above negative leading principle minors show that the $\mathbf{w}|_\mathcal{S}$ form the local minima of $f|_\mathcal{S}$.

In conclusion, only $\pm\mathbf{e}_i$'s are the local maximum of $f(\mathbf{w})$. This also means the local minima of $M_{D(\mathbf{C}),4}$ are only located at $\pm\mathbf{c}_i$'s. □

Now that we know that the secrets are exactly the local minima of the kurtosis, we can retrieve them by gradient descent. Note that we have an explicit expression of the gradient: for a unit vector $\mathbf{w}$, the gradient of $M_{D(\mathbf{C}),4}(\mathbf{w})$ is

$$\nabla M_{D(\mathbf{C}),4}(\mathbf{w}) = 12\mathbf{w} - \sum_{i=1}^n (12 - 4\alpha_i) \langle \mathbf{c}_i, \mathbf{w} \rangle^3 \mathbf{c}_i.$$

The descent is implemented as a geodesic flow, similarly to [TW20], which is significantly more efficient than the naive approach of [NR06]. Pseudocode is provided in Algorithm 3. We point out that the gradient is not known exactly so we need to approximate it using all the samples of S at each step, similarly to that the covariance of $D(\mathbf{B})$ is only known up to the best approximation allowed by S.

To sum up, after putting everything together, the general hidden transformation problem can be solved by Algorithm 4.

4.2 The Case of PEREGRINE

While Section 4.1 depicts the generic algorithmic framework for learning the hidden transformation, we now discuss the concrete attack for PEREGRINE in more details.

Algorithm 3: Descent

Input: A distribution D such that $\mathbf{Cov}[D] = \mathbf{I}_n$ and a set S of samples from $D(\mathbf{C})$ where $\mathbf{C}\mathbf{C}^t = \mathbf{I}_n$.
Output: An approximation of one column of $\pm\mathbf{C}$

1 Choose a random vector $\mathbf{w}$ uniformly over the unit sphere $\mathbb{S}^{n-1}$
2 Use the samples in S to compute (an approximation of) the gradient $\mathbf{g} = \nabla M_{D(\mathbf{C}),4}(\mathbf{w})$
3 $\mathbf{h} \leftarrow -\mathbf{g} + \langle \mathbf{g}, \mathbf{w} \rangle \mathbf{w}$
4 $\mathbf{h} \leftarrow \mathbf{h}/\|\mathbf{h}\|$
5 $\theta \leftarrow \theta_0$
6 **while** $\theta \geq \theta_{min}$ **do**
7 $\quad \mathbf{w}_{new} \leftarrow \mathbf{w} \cdot \cos\theta + \mathbf{h} \cdot \sin\theta$
8 $\quad \mathbf{w}_{new} \leftarrow \mathbf{w}_{new}/\|\mathbf{w}_{new}\|$
9 $\quad$ **if** $\left|M_{D(\mathbf{C}),4}(\mathbf{w}_{new}) - M_{D(\mathbf{C}),4}(\mathbf{w})\right| < \frac{1}{2} \cdot \theta \cdot \langle \mathbf{h}, \mathbf{g} \rangle$ **then**
10 $\quad\quad$ **goto** Step 10
11 $\quad$ **end if**
12 $\quad \theta \leftarrow \nu \cdot \theta$
13 **end while**
14 $\mathbf{w} \leftarrow \mathbf{w}_{new}$
15 **if** $\theta < \theta_0$ **then**
16 $\quad \theta_0 \leftarrow \theta/\nu$
17 $\quad$ **goto** Step 2
18 **end if**
19 **return** $\mathbf{w}$

Algorithm 4: Solver

Input: A distribution D and a set S of samples drawn from $D(\mathbf{B})$
Output: An approximation of one column of $\pm\mathbf{B}$

1 $\mathbf{L} \leftarrow \mathsf{Deform}(D, S)$
2 $S' \leftarrow \{\mathbf{L}\mathbf{s} \mid \mathbf{s} \in S\}$
3 $\mathbf{D} \leftarrow \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$
4 Let D' be the distribution of $\mathbf{D}^{-1}\mathbf{x}$ for $\mathbf{x} \sim D$
5 $\mathbf{w} \leftarrow \mathsf{Descent}(D', S')$
6 **return** $\sigma_i^{-1}\mathbf{L}^{-1}\mathbf{w}$ for i minimizing $\mathbb{E}_{z_i \leftarrow D'_i}[z_i^4]$

Exploiting the ring structure. PEREGRINE is built over $\mathbb{Z}[x]/(x^{n/2}+1)$ and such a ring structure can be used to greatly improve practical attacks [DN12]:

- We can generate $\frac{n}{2}$ additional samples from one signature transcript using the ring automorphisms, i.e. by multiplying x^i. This makes the approximations more accurate for the same amount of traces collected, for instance for the empirical covariance estimation, we have:

$$\mathbf{K} = \frac{2}{n\#S - 2} \sum_{\mathbf{x} \in S} \sum_{i=0}^{n/2-1} \mathrm{vec}(x^i \cdot \mathbf{x})\mathrm{vec}(x^i \cdot \mathbf{x})^t$$

  where vec denotes the vector representation of the polynomial.
- All involved matrices (i.e. $\mathbf{\Sigma}$, $\mathbf{B}$ and $\mathbf{K}$) are now 2-by-2 over $\mathcal{K}_{\mathbb{R}} = \mathbb{R}[x]/(x^{n/2}+1)$, which allows to make the computation of the Gram root $\mathbf{P}$ much faster. In particular, our practical attacks leverage the Denman–Beavers iteration [DJ76], *at the ring level* to compute an approximate $\mathbf{P}$ over $\mathcal{K}_{\mathbb{R}}^{2\times 2}$ in *quasi-linear time* instead of requiring full rounds of linear algebra (which would requires at least $O(n^\omega)$ time, with ω the exponent of matrix multiplication).

We now derive the formulas for the fourth moment and its gradient for both versions of PEREGRINE, namely the one appearing in the specification and the one in the implementation code.

General computation. In the general setting considered at the start of this section, where D is the product of centered independent distributions D_i of standard deviations σ_i not necessarily equal to 1, the discussion of Section 4.1.2 shows that the fourth moment and its gradient have the following expressions:

$$M_{D(\mathbf{C}),4}(\mathbf{w}) = 3\|\mathbf{w}\|^4 - \sum_{i=1}^{n}(3-\alpha_i)\langle \mathbf{c}_i, \mathbf{w}\rangle^4,$$
$$\nabla M_{D(\mathbf{C}),4}(\mathbf{w}) = 12\mathbf{w} - \sum_{i=1}^{n}(12-4\alpha_i)\langle \mathbf{c}_i, \mathbf{w}\rangle^3 \mathbf{c}_i.$$

where $\alpha_i = \mathbb{E}[z_i^4]$, for z_i distributed according to $\frac{1}{\sigma_i} \cdot D_i$. In particular, if we denote by $\mu_{4,i}$ the fourth moment of D_i, we have $\alpha_i = \mu_{4,i}/\sigma_i^4$.

For the distributions D_i of relevance to PEREGRINE, we can easily compute those values α_i using moment generating functions. In all cases, D_i is a convolution $D_i = U\big([-1/2, 1/2)\big) + B_\mu$ for some $\mu \geq 0$. Now the moment generating function of $U\big([-1/2, 1/2)\big)$ is given by:

$$\mathrm{MGF}_{U([-1/2,1/2))}(t) = \frac{e^{t/2} - e^{-t/2}}{t} = \frac{\sinh(t/2)}{t/2},$$

and that of B_μ for any even $\mu \geq 0$ is given by:

$$\mathrm{MGF}_{B_\mu}(t) = \Big(\frac{e^{t/2} + e^{-t/2}}{2}\Big)^\mu = \cosh(t/2)^\mu.$$

As a result:

$$\mathrm{MGF}_{D_i}(t) = \frac{\sinh(t/2) \cdot \cosh(t/2)^\mu}{t/2}$$

by multiplicativity in convolutions. On the other hand, by definition:

$$\mathrm{MGF}_{D_i}(t) = 1 + \frac{\sigma_i^2}{2!} \cdot t^2 + \frac{\mu_{4,i}}{4!} \cdot t^4 + o(t^4).$$

Thus, it suffices to obtain a 4th order Taylor expansion of (4.2) to compute α_i.

Specification version. According to the specification parameters,

$$D_i = U\big([-1/2, 1/2)\big) + B_\mu \quad \text{with} \quad \mu = 26$$

for all $1 \leq i \leq n$. Thus:

$$\mathrm{MGF}_{D_i}(t) = \frac{\sinh(t/2) \cdot \cosh(t/2)^{26}}{t/2} = 1 + \frac{79}{24} \cdot t^2 + \frac{10141}{1920} \cdot t^4 + o(t^4).$$

As a result:

$$\alpha_i = \frac{4! \cdot 10141/1920}{(2! \cdot 79/24)^2} = \frac{91269}{31205}$$

for all $1 \leq i \leq n$. Hence, we obtain the corresponding fourth moment function as:

$$M_{D(\mathbf{C}),4}(\mathbf{w}) = 3\|\mathbf{w}\|^4 - \frac{2346}{31205}\sum_{i=1}^{n}\langle \mathbf{c}_i, \mathbf{w}\rangle^4$$

and its gradient as:

$$\nabla M_{D(\mathbf{C}),4}(\mathbf{w}) = 12\mathbf{w} - \frac{9384}{31205}\sum_{i=1}^{n}\langle \mathbf{c}_i, \mathbf{w}\rangle^3 \mathbf{c}_i.$$

Reference implementation version. The reference implementation uses more aggressive parameters: the binomial parameters $(\mu_1, \mu_2) = (6, 0)$. Hence,

$$D_i = \begin{cases} U\big([-1/2, 1/2)\big) + B_\mu & \text{with} \quad \mu = 6 \quad \text{for } 1 \le i \le n/2; \\ U\big([-1/2, 1/2)\big) & \text{for } n/2 + 1 \le i \le n. \end{cases}$$

For $1 \le i \le n/2$, we therefore get:

$$\mathrm{MGF}_{D_i}(t) = \frac{\sinh(t/2) \cdot \cosh(t/2)^6}{t/2} = 1 + \frac{19}{24} \cdot t^2 + \frac{541}{1920} \cdot t^4 + o(t^4)$$

and hence:

$$\alpha_i = \frac{4! \cdot 541/1920}{(2! \cdot 19/24)^2} = \frac{4869}{1805}.$$

On the other hand, for $n/2 + 1 \le i \le n$, we simply have:

$$\mathrm{MGF}_{D_i}(t) = \frac{\sinh(t/2)}{t/2} = 1 + \frac{1}{24} \cdot t^2 + \frac{1}{1920} \cdot t^4 + o(t^4)$$

and hence:

$$\alpha_i = \frac{4!/1920}{(2!/24)^2} = \frac{9}{5}.$$

We therefore obtain the fourth moment function and its gradient for this version as:

$$M_{D(\mathbf{C}),4}(\mathbf{w}) = 3\|\mathbf{w}\|^4 - \frac{546}{1805} \sum_{i=1}^{n/2} \langle \mathbf{c}_i, \mathbf{w} \rangle^4 - \frac{6}{5} \sum_{i=n/2+1}^{n} \langle \mathbf{c}_i, \mathbf{w} \rangle^4,$$

$$\nabla M_{D(\mathbf{C}),4}(\mathbf{w}) = 12\mathbf{w} - \frac{2184}{1805} \sum_{i=1}^{n/2} \langle \mathbf{c}_i, \mathbf{w} \rangle^3 \mathbf{c}_i - \frac{24}{5} \sum_{i=n/2+1}^{n} \langle \mathbf{c}_i, \mathbf{w} \rangle^3 \mathbf{c}_i.$$

5 Practical Key Recovery Attack against PEREGRINE

By combining the statistical learning in Section 4 and the trick of [Pre23], we propose a full key recovery attack against PEREGRINE–512. For both specification and reference implementation versions, our attack is able to fully recover the secret key of PEREGRINE in practice. In this section, we present the relevant implementation details and collect the experimental results of our attacks.

5.1 Gradient Descent

As discussed in the previous section, our attack uses gradient descent to find an approximation of secret vectors. The original gradient descent in [NR06] computes a step δ for the iteration where $\mathbf{w}_{new} = \mathbf{w} - \delta\mathbf{g}$ and $\mathbf{g}$ is the gradient $\nabla M_{D(\mathbf{C}),4}(\mathbf{w})$. This converges slowly and does not behave well in our case. To overcome this limitation, we instead implement the gradient descent following the one in [TW20], as described in Algorithm 3. This algorithm takes adaptive steps along geodesic paths on the unit sphere to search for a local minimum of $M_{D(\mathbf{C}),4}(\mathbf{w})$ restricted to the sphere, until $\|\mathbf{h}\|$ becomes sufficiently small. This improves the speed of convergence and proves quite effective in our setting. For simplicity's sake, our attack uses the same gradient descent parameters as in [TW20], namely $\theta_0 = 0.25$, $\theta_{min} = 0.005$ and $\nu = 0.8$.

5.2 Correcting Approximate Errors with Lattice Decoding

Once it converges, the gradient descent of Algorithm 4 outputs a vector which essentially follows a normal distribution centered at one of the basis vectors. Equivalently, it can be expressed as one of the basis vector plus some normal error vector, whose magnitude depends on the number of signatures in the attack. In order to recover the basis vector and actually break the scheme, a post-processing step is needed to correct the error.

The Nguyen–Regev decoding techniques. In their original attack against GGH and NTRUSign [NR06], Nguyen and Regev mentioned two possible approaches to carry out this post-processing. The first approach is simply to hope that the error becomes less than $1/2$ in infinity norm, so that the basis vector can be directly obtained by coefficient-wise rounding. This is simple but typically requires a relatively large number of signatures. The second approach is to use lattice reduction: the search for the target basis vector is viewed as a bounded distance decoding problem in the public lattice, which can be solved using standard techniques (Nguyen and Regev suggested applying Babai's nearest plane algorithm after reducing the public basis with BKZ; one could also reduce to unique SVP with Kannan's embedding technique and try to solve that unique SVP instance with BKZ).

The same two approaches can also be applied in our setting. However, due to the added noise in PEREGRINE, the magnitude of the error is substantially larger than in GGH and NTRUSign (especially in the "specification" version of PEREGRINE), and lattice dimensions are much larger than those considered in [NR06]. This tends to make lattice reduction-based decoding fairly impractical: the complexity of lattice reduction attacks, while it quickly falls below the claimed security level of the scheme itself, tends to remain too large to carry out the decoding in practice right up until the point where coefficient-wise rounding becomes feasible[13]. This leaves component-wise rounding, possibly complemented by "meet-in-the-middle" type combinatorial improvements, as the better suited approach in our case, and it requires lots of signatures to succeed.

Prest's trick. Fortunately, since PEREGRINE is an NTRU-based scheme, one can do better than either of the Nguyen–Regev decoding techniques. As pointed out by Prest in [Pre23], it suffices to correctly recover *half* of the coefficients of a basis vector to break the scheme, since the remaining half can be deduced from the first using the NTRU equation (in much the same way as signature compression works in those schemes).

Formally, let $\mathbf{b} = (b^{(1)}, b^{(2)}) \in \mathcal{L}_{\text{NTRU}}$ be the target vector to be recovered and $\mathbf{b}' = \left((b')^{(1)}, (b')^{(2)}\right)$ be the approximation output by Algorithm 4. Suppose that we know at least $n/2$ indices i such that the rounding $\lfloor \mathbf{b}'_i \rceil$ matches the corresponding coefficient $\mathbf{b}_i$ of $\mathbf{b}$. This means that the difference $\mathbf{d} = \lfloor \mathbf{b}'_i \rceil - \mathbf{b} = (d^{(1)}, d^{(2)})$ has zeros in at least $n/2$ known positions. Now we have $b^{(1)} + b^{(2)} \cdot h = 0 \bmod q$ by definition of the NTRU lattice, and therefore:

$$\left\lfloor (b')^{(1)} \right\rceil + \left\lfloor (b')^{(2)} \right\rceil \cdot h = d^{(1)} + d^{(2)} \cdot h \bmod q.$$

This is an equation of polynomials of degree $n/2$, and we know the entire left-hand side; moreover, there are at most $n/2$ unknowns on the right-hand side. Therefore, we can solve the linear system for $\mathbf{d}$ (which is of course exactly revealed by its remainder modulo q since it is small) and hence recover $\mathbf{b}$ exactly.

In order to apply this idea, one then needs to determine a subset of at least $n/2$ coefficients of $\mathbf{b}'$ that are likely to be correctly rounded to the corresponding coefficient of $\mathbf{b}$. In [Pre23], Prest does so by selecting a certain threshold $\varepsilon \in (0, 1/2)$ and regarding a coefficient of $\mathbf{b}'$ as correctly rounded when its fractional part doesn't exceed ε.

We proceed in a slightly different way, which bypasses the need to select a threshold value, and which has some additional benefits discussed later in this section: our approach is simply to select the $n/2$ coefficients that have the highest probability of rounding correctly.

Rounding the best half of the coefficients. As discussed earlier, by standard properties of the gradient descent, we can model the vector $\mathbf{b}'$ as a normal vector centered around $\mathbf{b}$. In particular, any fixed coefficient b' of $\mathbf{b}'$ follows a normal distribution $\mathcal{N}(b, \sigma^2)$ centered at the corresponding coefficient of $\mathbf{b}$, and of a certain variance σ^2 (not *a priori* the same for all coefficients).

Then, we claim that the probability of correct rounding (i.e., the probability that $\lfloor b' \rceil = b$) is a function of the fractional part $x = b' - \lfloor b' \rceil$ (which is a known value) and of σ (which we assume can be estimated in our setting). It can be expressed as follows.

Lemma 4. *Let $b' \sim \mathcal{N}(b, \sigma^2)$ for some unknown* integer *center b, and known standard deviation σ. Let $x = b' - \lfloor b' \rceil$. The probability that $\lfloor b' \rceil = b$ is given by:*

$$\psi_\sigma(x) = \frac{\rho_\sigma(x)}{\rho_\sigma(x + \mathbb{Z})}$$

[13] For the practical exploitation we propose, the required blocksize would be of the order of 200, which is untractable on a reasonnable machine and orders of magnitude larger than the simple exploitation via linear algebra.

where we let as usual $\rho_\sigma(t) = \exp\big(-t^2/(2\sigma^2)\big)$.

Proof. Let $e = b' - b \sim \mathcal{N}(0, \sigma^2)$. Then $e - x = \lfloor b' \rceil - b$ is an integer, and is zero exactly when $\lfloor b' \rceil = b$. Therefore, the probability $\psi_\sigma(x)$ of correct rounding is given by:

$$\psi_\sigma(x) = \Pr\big[e = x \big| e \in x + \mathbb{Z}\big] = \frac{\rho_\sigma(x)}{\rho_\sigma(x + \mathbb{Z})}$$

since ρ_σ is up to a constant factor the probability density function of the normal distribution $\mathcal{N}(0, \sigma^2)$ of e. □

Now, let us first analyze our approach in a simple model where the distribution of the entire vector $\mathbf{b}'$ is a spherical Gaussian $\mathcal{N}\big(\mathbf{b}, \sigma^2 \cdot \mathbf{I}_n\big)$ of known standard deviation σ. In particular, the correct rounding probability p_i of the i-th coefficient is given by $\psi_\sigma(x_i)$ for all i, with $\mathbf{x} = (x_1, \ldots, x_n) = \mathbf{b}' - \lfloor \mathbf{b}' \rceil$. Furthermore, all the coefficients are independent; therefore, if we denote by:

$$0 \leq p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(n)} \leq 1$$

the probabilities p_i sorted in increasing order, the probability that the $n/2$ coefficients of $\mathbf{b}'$ with the highest probability of correct rounding are in fact correctly rounded is given by:

$$p_{\text{succ}} = \prod_{k=n/2+1}^{n} p_{(k)}.$$

In order to analyze our probability of success, we can estimate the expectation of $\log p_{\text{succ}}$ (which is somewhat better behaved than p_{succ} itself). To that end, we show that the sum of the top half (or any constant fraction) of the order statistics of n independent and identically distributed random variables has a relatively convenient expression as follows.

Lemma 5. *Let $X_1, \ldots, X_n$ be independent and identically distributed real-valued random variables with probability density function f_X and CDF F_X. Denote by $X_{(1)} \leq \cdots \leq X_{(n)}$ the corresponding values in non-decreasing order (the order statistics). Fix furthermore some constant $\alpha \in (0, 1)$ and let:*

$$E_{n,\alpha} = \mathbb{E}\Big[\sum_{k > \alpha n} X_{(k)}\Big].$$

Then the following asymptotic equivalence holds:

$$E_{n,\alpha} \underset{n \to +\infty}{\sim} n \int_{F_X^{-1}(\alpha)}^{+\infty} x \cdot f_X(x)\, dx = n \int_\alpha^1 F_X^{-1}(u)\, du.$$

Proof. The well-known expression of the probability density function of $X_{(k)}$ is given by (see e.g. [DN03, Eq. (2.1.6)]):

$$f_{(k)}(x) = \frac{n!}{(k-1)!(n-k)!} f_X(x) \cdot F_X(x)^{k-1} \big(1 - F_X(x)\big)^{n-k}.$$

Thus, we get:

$$\begin{aligned} E_{n,\alpha} &= \int_{-\infty}^{+\infty} x \cdot \frac{f_X(x)}{F_X(x)} \sum_{k > \alpha n} k \binom{n}{k} F_X(x)^k \big(1 - F_X(x)\big)^{n-k}\, dx \\ &= \int_{-\infty}^{+\infty} x \cdot f_X(x) \cdot n \varphi_{n,\alpha}\big(F_X(x)\big)\, dx \end{aligned}$$

where we have let:

$$\varphi_{n,\alpha}(p) = \frac{1}{np} \sum_{k > \alpha n} k \binom{n}{k} p^k (1-p)^{n-k}.$$

In particular, $\varphi_{n,\alpha}(p) = \frac{1}{p} \mathbb{E}\big[\frac{1}{n} Z \mid \frac{1}{n} Z > \alpha\big]$ for $Z \sim \text{Binomial}(n, p)$. Now $\frac{1}{n} Z$ concentrates rapidly around p for large n. Therefore $\frac{1}{n} Z > \alpha$ happens with probability very close to 1 if $p > \alpha$, and very close to 0 if

$p < \alpha$. It easily follows that $\varphi_{n,\alpha}(p)$ converges pointwise to 0 for $p < \alpha$ and to $\frac{1}{p}\mathbb{E}\big[\frac{1}{n}Z\big] = p/p = 1$ for $p > \alpha$. A dominated convergence arguments then yields:

$$E_{n,\alpha} \sim n \int_{-\infty}^{+\infty} x \cdot f_X(x) \cdot [\![F_X(x) > \alpha]\!]\, dx = n \int_{F_X^{-1}(\alpha)}^{+\infty} x \cdot f_X(x)\, dx.$$

Making the change of variables $u = F_X(x)$, hence $x = F_X^{-1}(u)$, $du = f_X(x)\, dx$, we also obtain:

$$E_{n,\alpha} \sim n \int_{\alpha}^{1} F_X^{-1}(u)\, du$$

as required.

Based on the above, we conclude that

$$\mathbb{E}\left[\log p_{\text{succ}}\right] = \mathbb{E}\Big[\sum_{k>n/2} \log p_{(k)}\Big] \underset{n\to+\infty}{\sim} n \int_{1/2}^{1} F^{-1}(u)\, du$$

where F is the common CDF of the $\log p_i = \log \psi_\sigma(x_i)$. We can compute, for all $t \le 0$:

$$F(t) = \Pr\left[\log \psi_\sigma(x_i) \le t\right] = \Pr\left[\psi_\sigma(x_i) \le e^t\right] = \Pr\left[|x_i| \ge \psi_\sigma^{-1}(e^t)\right]$$

since ψ_σ is decreasing on $[0, 1/2]$ and even (and $x_i \in [-1/2, 1/2)$). Now if we again let $e_i = b'_i - b_i \sim \mathcal{N}(0, \sigma^2)$, we have $x_i = e_i - \lfloor e_i \rceil$, so that:

$$\begin{aligned} F(t) &= \Pr\left[\left|e_i - \lfloor e_i \rceil\right| \ge \psi_\sigma^{-1}(e^t)\right] = 1 - 2\Pr\left[0 \le \left|e_i - \lfloor e_i \rceil\right| < \psi_\sigma^{-1}(e^t)\right] \\ &= 1 - 2\Pr\left[e_i \in \left[0, \psi_\sigma^{-1}(e^t)\right) + \mathbb{Z}\right] = 1 - \sum_{k\in\mathbb{Z}} \operatorname{erf}\left(\frac{\psi_\sigma^{-1}(e^t) + k}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{k}{\sigma\sqrt{2}}\right). \end{aligned}$$

This expression is somewhat cumbersome, but our attack approach is only workable when σ is small compared to 1, in which case all the terms in the sum over k are actually negligible except for $k = 0$. Therefore, for small σ, the following approximation holds:

$$F(t) \approx 1 - \operatorname{erf}\left(\frac{\psi_\sigma^{-1}(e^t) + k}{\sigma\sqrt{2}}\right).$$

And as a result, by inverting this formula we similarly obtain:

$$F^{-1}(u) \approx \log \psi_\sigma\big(\sigma\sqrt{2} \cdot \operatorname{erf}^{-1}(1-u)\big).$$

Finally, recall that:

$$\psi_\sigma(x) = \frac{\rho_\sigma(x)}{\rho_\sigma(x+\mathbb{Z})} = \frac{\exp\left(-\frac{x^2}{2\sigma^2}\right)}{\sum_{k\in\mathbb{Z}} \exp\left(-\frac{(x-k)^2}{2\sigma^2}\right)} = \frac{1}{\sum_{k\in\mathbb{Z}} \exp\left(\frac{2kx-k^2}{2\sigma^2}\right)}.$$

For σ small compared to 1, most of the terms in the sum in the denominator are negligibly small. Keeping only the terms $k \in \{-1, 0, 1\}$ provides a very close approximation in the range of interest. Hence:

$$\psi_\sigma(x) \approx \frac{1}{1 + \exp\left(\frac{2x-1}{2\sigma^2}\right) + \exp\left(\frac{-2x-1}{2\sigma^2}\right)} = \frac{1}{1 + 2\exp(-1/2\sigma^2)\cosh(x/\sigma^2)}.$$

Putting all together, we finally obtain the following approximation of $\mathbb{E}\left[\log p_{\text{succ}}\right]$, valid for large n and σ small compared to 1:

$$\mathbb{E}\left[\log p_{\text{succ}}\right] \approx -n \int_{1/2}^{1} \log\left(1 + 2e^{-1/2\sigma^2} \cosh\left(\frac{\sqrt{2}}{\sigma} \operatorname{erf}^{-1}(1-u)\right)\right) du$$

(which can be simplified a bit further with the change of variables $u \mapsto 1-u$). This expression is convenient for numerical computation, and lets us obtain the plot of $\exp \mathbb{E}\left[\log p_{\text{succ}}\right]$ (which by Jensen's inequality is a lower bound of $\mathbb{E}\left[p_{\text{succ}}\right]$) according to σ given in Fig. 1.

Modeling the Gaussian error. As discussed above, a somewhat simple model of the error $\mathbf{e} = \mathbf{b}' - \mathbf{b}$ is that it is distributed as a spherical Gaussian vector $\mathcal{N}(0, \sigma^2 \cdot \mathbf{I}_n)$ of standard deviation σ depending on the number N of signatures used in the attack. As usual, we more precisely expect the standard deviation to decrease proportionally[14] to $\sqrt{N}$. In this model, we thus have $\sigma \approx C_\sigma/\sqrt{N}$ for some constant C_σ that can be derived experimentally by curve fitting.

We for example find that the size of the error in the attack against the specification version of PEREGRINE–512 is well-described by this model with $C_\sigma \approx 840$ (see Section 5.3 below). Plugging back the corresponding value of σ in our estimate of the probability of success p_{succ}, we obtain the graph in Fig. 2 of $\exp\left(\mathbb{E}\left[\log p_{\text{succ}}\right]\right)$ as a function of N, which already captures the behavior of the attack fairly accurately as will be shown in Section 5.3.

However, it is not entirely accurate to model the error vector $\mathbf{e}$ as a spherical Gaussian. Indeed, the gradient descent is not directly carried out with respect to the original basis $\mathbf{B}$ of which $\mathbf{b}$ is a vector, but with respect to the deformed basis $\mathbf{LB}$. Therefore, a more accurate model describes the output $\mathbf{w}$ of the Descent algorithm as a spherical[15] Gaussian vector of covariance $(\sigma')^2 \cdot \mathbf{I}_n$, whereas the recovered vector $\mathbf{b}'$ output by Solver is given by an expression of the form $\mathbf{b}' = \mathbf{L}_0^{-1}\mathbf{w}$. As a result, the error $\mathbf{e} = \mathbf{b}' - \mathbf{b}$ is indeed Gaussian, but with a known covariance matrix $\mathbf{\Sigma} = (\sigma')^2 \cdot \mathbf{L}_0^{-1}(\mathbf{L}_0^{-1})^t$ which is no longer scalar in general. As before, σ' decreases like the square root of the number of signatures: we have another constant $C_{\sigma'}$, derived experimentally, such that $\sigma' \approx C_{\sigma'}/\sqrt{N}$.

In this more accurate model, each of the coefficients e_i of $\mathbf{e}$ remain normally distributed, but with potentially distinct[16] standard deviations σ_i. We therefore use those values σ_i to compute the probabilities of correct rounding $\psi_{\sigma_i}(x_i)$ of the various coefficients, and still keep the top half most likely to be correctly rounded when applying Prest's trick.

This provides a practical improvement in our experiments compared to the simpler model when all the σ_i's are assumed to be equal. This holds even though we still ignore the fact that the covariance $\mathbf{\Sigma}$ need not be diagonal, so that there the various coefficients may not be independent. It might be possible to improve the attack further by leveraging the entire covariance matrix $\mathbf{\Sigma}$ instead of only its diagonal coefficients, but we have not found a practical way to do so yet.

5.3 Experimental Results

We implemented the aforementioned key recovery attack against PEREGRINE. The statistical learning part is implemented in Rust using f64 precision and some parallelizations are used to compute the covariance, kurtosis and the gradient descent. The decoding part is implemented in Python. The code of the attack can be found at `https://github.com/lxhcrypto/Peregrine_attack`.

We validated the attack on both the reference implementation and the specification version of PEREGRINE–512. We performed experiments over 10 instances and for each sample size N, we randomly chose 5 starting points in the gradient descent. For each instance and sample size, the attack is counted a success if it recovers the full secret key via decoding at least one approximation output by the statistical learning algorithm with some starting point.

Based on experimental measures and curve fitting in the original basis (resp. the deformed basis), we get the following estimates for the constants C_σ and $C_{\sigma'}$ describing the respective magnitude of the error vectors:

- $C_\sigma \approx 840$ and $C_{\sigma'} \approx 1.4$ for the specification version;
- $C_\sigma \approx 41$ and $C_{\sigma'} \approx 0.15$ for the reference implementation.

The curves are shown in Figure 3.

[14] In the Nguyen–Regev simple version of the gradient descent, the fact that the standard deviation is $\propto N^{-1/2+o(1)}$ is a consequence of the analysis of [NR06, Sec. 6]. The Tibouchi–Wallet gradient descent [TW20] is trickier to analyze, but expectedly follows the same asymptotic behavior in experiments.

[15] In fact, since the gradient descent is carried out on the unit sphere, it would be even more correct to model $\mathbf{w}$ as a Gaussian on the sphere, seen as a Riemannian manifold, so that the corresponding error is essentially a centered spherical Gaussian not in the whole space but on the tangent hyperplane. We ignore this further refinement in what follows.

[16] In fact, the symmetry provided by the ring structure ensures that there are at most two distinct values of the σ_i's, and those two values are moreover fairly close in the specification version of PEREGRINE. They are however markedly different in the reference implementation version, due to the two different binomial parameters involved.

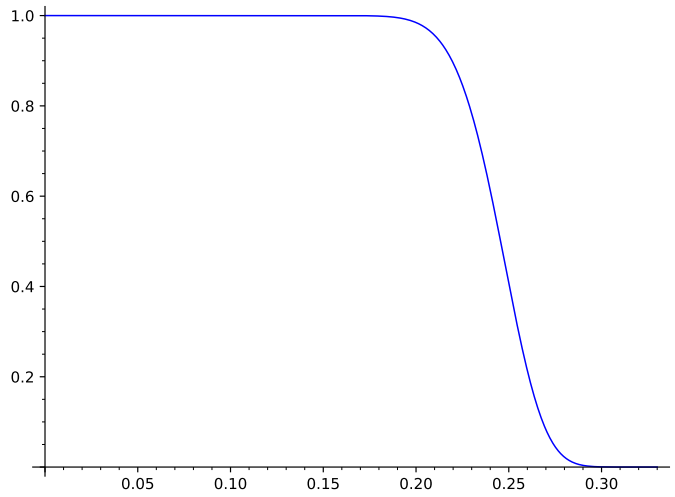

Fig. 1. Expected success probability (more precisely $\exp\left(\mathbb{E}\left[\log p_{\text{succ}}\right]\right) \leq \mathbb{E}\left[p_{\text{succ}}\right]$) of our decoding approach for $n = 1024$ (i.e., PEREGRINE–512), assuming a spherical Gaussian error of standard deviation σ.

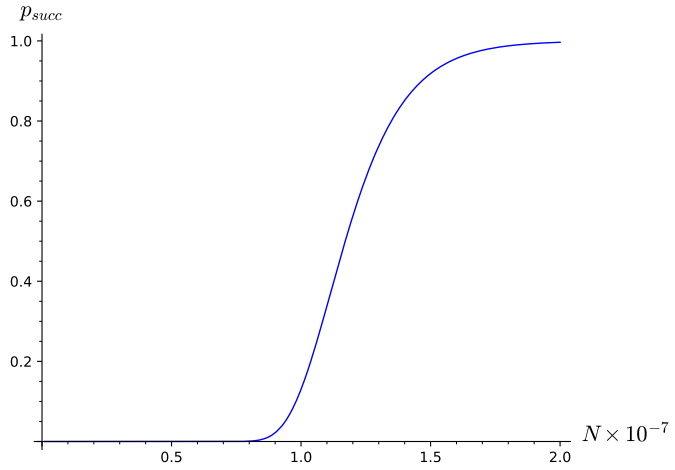

Fig. 2. Expected success probability (more precisely $\exp\left(\mathbb{E}\left[\log p_{\text{succ}}\right]\right)$) of our decoding approach for the attack on the specification version of PEREGRINE–512, as a function of the number N of signatures.

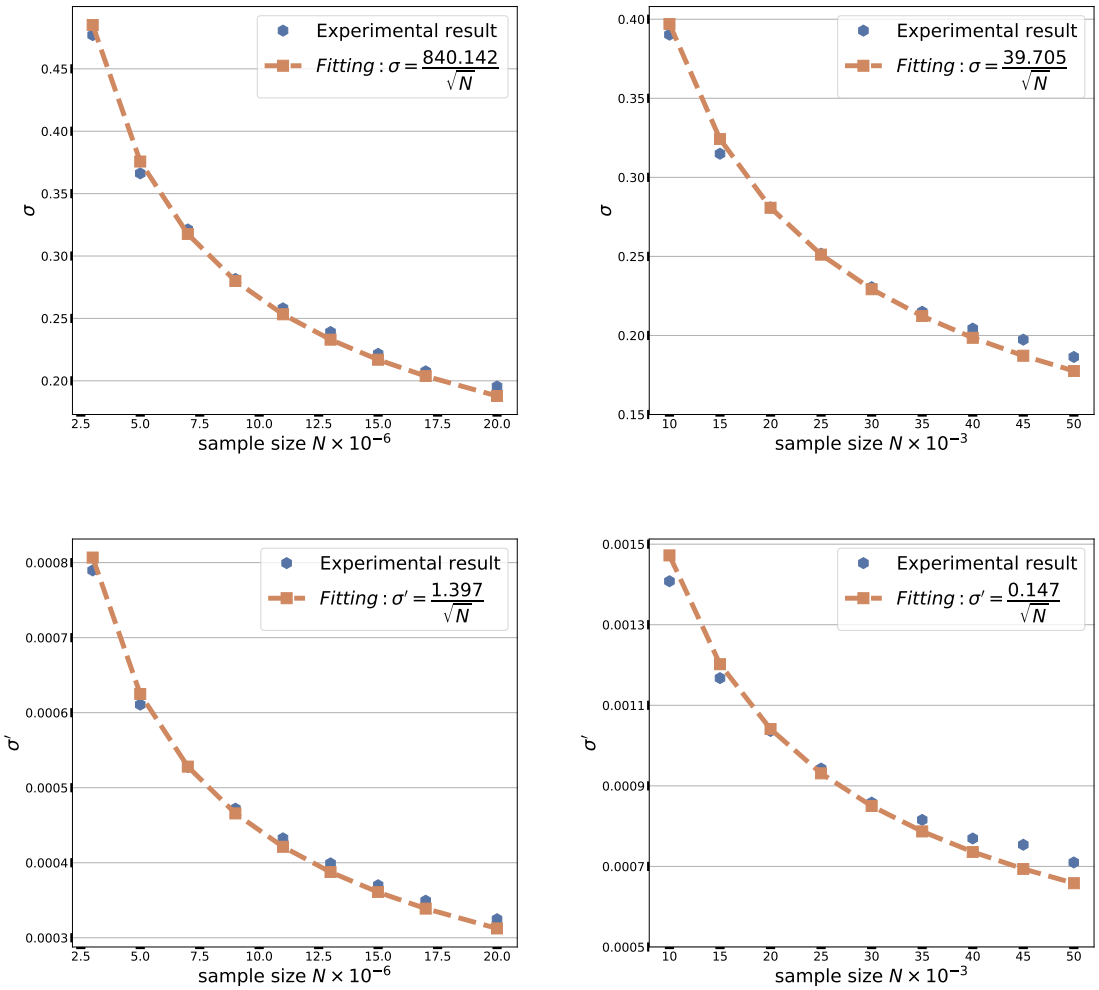

Fig. 3. Experimental measure of the constants C_σ (above) and $C_{\sigma'}$ (below) by curve fitting. The left-hand graphs are for the specification version and the right-hand ones for the reference implementation version. Experimental values measure over 10 PEREGRINE–512 instances and for each sample size, we use 5 random starting points in the gradient descent.

Table 2. The number of the successful key recovery on the reference implementation for each instances and all 5 starting points.

$N \times 10^{-3}$	10	15	20	25	30	35	40	45	50
Instance 1	0	0	0	2	4	5	5	4	5
Instance 2	0	0	1	1	5	3	5	5	5
Instance 3	0	0	2	3	3	4	5	5	5
Instance 4	0	0	0	0	5	5	5	5	5
Instance 5	0	0	0	3	1	5	5	5	5
Instance 6	0	0	0	3	5	5	5	5	5
Instance 7	0	0	0	1	4	4	5	5	5
Instance 8	0	0	0	3	5	3	5	5	5
Instance 9	0	0	0	0	5	5	5	5	5
Instance 10	0	0	0	4	2	5	5	5	5

Table 3. The number of the successful key recovery on the specification version for each instances and all 5 starting points.

$N \times 10^{-6}$	3	5	7	9	11	13	15	17	20
Instance 1	0	0	0	0	3	5	5	5	5
Instance 2	0	0	0	0	0	4	3	5	5
Instance 3	0	0	0	0	0	5	5	5	5
Instance 4	0	0	0	0	0	2	4	5	5
Instance 5	0	0	0	0	3	3	3	5	5
Instance 6	0	0	0	0	1	5	5	5	5
Instance 7	0	0	0	0	4	3	5	5	5
Instance 8	0	0	0	0	2	3	2	5	5
Instance 9	0	0	0	0	0	5	5	5	5
Instance 10	0	0	0	0	3	3	5	5	5

Key recovery attack against the reference implementation. The attack on the reference implementation is highly efficient. When $N \gtrsim 50{,}000$, one can get a good approximation $\mathbf{b}'$ such that $\|\lfloor \mathbf{b}' \rceil - \mathbf{b}\|_1 \leq 7$ with a good probability, which allows to completely recover the key by a simple exhaustive search within half an hour. In order to further reduce the number of required signatures, one can employ the decoding technique in Table 2 and give a successful key recovery with 20,000 signatures on 2 out of 10 instances, and with 25,000 signatures on 8 out of 10 instances.

Key recovery attack against the specification. The attack on the specification version is more costly, as the specification uses larger binomial parameters. It now requires around 20 million signatures to succeed in key recovery by a simple exhaustive search within half an hour. By resorting to the decoding technique of Section 5.2, we achieve a successful attack with 11 million signatures on 6 out of 10 instances, as shown in Table 3. A visualization of the behavior of the decoding technique is provided in Fig. 4 (on the first 5 instances so as not to crowd the graph): each color represents a different instance, and each data point shows, for a particular run of the gradient descent, the predicted probability of success $p_{\text{succ}} = \prod_{k=n/2+1}^{n} \psi_{\sigma_i}(x_i)$ as well as the actuall outcome (success or failure). As we can see, the results closely match the model depicted in Fig. 2.

6 Conclusion and Perspectives

We have shown that the PEREGRINE signature scheme is vulnerable to a practical statistical attack. More concretely, for the reference implementation (resp. the specification) of PEREGRINE–512, we completely recover the signing key within a few hours, provided that around 25,000 (resp. 11 million) signatures samples are available. The same attack is expected to break the PEREGRINE–1024 parameters as well.

The design weakness of PEREGRINE revealed by our attack does not seem easy to fix. For secure hash-and-sign lattice signatures, the signature distribution is supposed to be simulatable without knowing the signing key. Currently, there are two main approaches to do so: Gaussian sampling due to [GPV08] and rejection sampling due to [LW15]. Adapting PEREGRINE to either of those approaches would negate its claimed efficiency advantage over schemes like Falcon. At the same time, continuing to pursue a strategy of heuristic countermeasures without provable security guarantees appears increasingly hopeless.

This is especially the case as our attack can be further improved in various ways, which we leave as possible avenues for future work:

- in our experiments, we only carry out the attack on 5 random starting points per key instance. We could easily increase our success rate (and hence decrease the required number of signatures) by simply retrying with more random starting points, as already noted in [NR06];
- moreover, up to negacyclic rotation and sign flips, there are only two equivalence classes of basis vectors in the NTRU basis. As a result, given k starting points in our attack, we expect to collect around $k/2$ approximation of each of those two basis vectors. Averaging out or using majority vote is thus expected to reduce the standard deviation of the error by a factor $\sqrt{k/2}$, and hence significantly improve the success rate. We only had time to carry out partial experiments with that idea, but we could already confirm that it consistently recovers the key in the specification version of PEREGRINE–512 with 9 million signatures samples;
- similarly, even when applying the attack with a single starting point, we can use the methodology of Section 5.2 to estimate the expected number of incorrectly rounded coefficient among the "best half" (in the specification version of PEREGRINE–512, it is e.g. ≈ 3.8 for 9 million signatures and ≈ 7.1 for 8 million). Decoding those remaining errors using combinatorial techniques (ranging from a simple exhaustive search to basic meet-in-the-middle to advanced algorithms like those of May [May21] and Kirshanova–May [KM21]) should therefore let us reduce the required number of signatures for full recovery significantly;
- on the implementation side, the limiting factor in our experiments is the speed of gradient descent in our relatively basic CPU implementation. We expect that a GPU implementation would greatly increase performance. Also, tuning the parameters of the descent algorithm of [TW20], although time consuming, may provide significant speed-ups as well.

References

Bab86. László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

DN03. Herbert A. David and Haikady N. Nagaraja. *Order Statistics*. Wiley, 3rd edition, 2003.

DJ76. E. D. Denman and A. N. Beavers Jr. The matrix sign function and computations in systems. *Journal of Applied Mathematics and Computation*, 2:63–94, 1976.

DLP14. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 22–41. Springer, Heidelberg, December 2014.

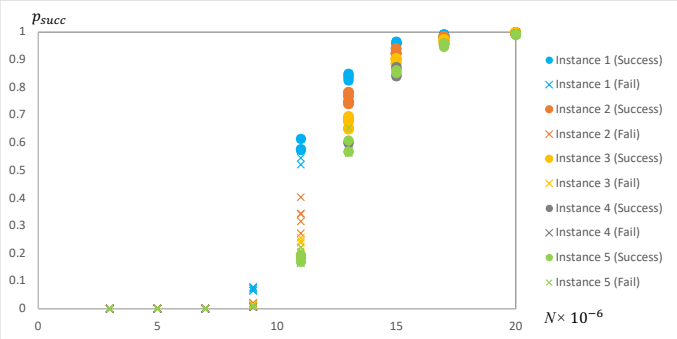

Fig. 4. Visualization of the "best half" decoding technique on the attack against the specification version of PEREGRINE–512.

DN12. Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 433–450. Springer, Heidelberg, December 2012.

EFG+22. Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 222–253. Springer, Heidelberg, May / June 2022.

ENS+23. Thomas Espitau, Thi Thu Quyen Nguyen, Chao Sun, Mehdi Tibouchi, and Alexandre Wallet. Antrag: Annular NTRU trapdoor generation. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT*, 2023. To appear.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

GS02. Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 299–320. Springer, Heidelberg, April / May 2002.

GGH97. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 112–131. Springer, Heidelberg, August 1997.

HHP+03. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. Springer, Heidelberg, April 2003.

KTW+22. Kwangjo Kim, Mehdi Tibouchi, Alexandre Wallet, Thomas Espitau, Akira Takahashi, Yang Yu, and Sylvain Guilley. SOLMAE: Submission to the Korea post-quantum cryptography competition round 1. Technical report, 2022. available at https://www.kpqc.or.kr/competition.html.

KM21. Elena Kirshanova and Alexander May. How to find ternary LWE keys using locality sensitive hashing. In Maura B. Paterson, editor, *18th IMA International Conference on Cryptography and Coding*, volume 13129 of *LNCS*, pages 247–264. Springer, Heidelberg, December 2021.

LDK+22. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

LW15. Vadim Lyubashevsky and Daniel Wichs. Simple lattice trapdoor sampling from a broad class of distributions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 716–730. Springer, Heidelberg, March / April 2015.

May21. Alexander May. How to meet ternary LWE keys. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 701–731, Virtual Event, August 2021. Springer, Heidelberg.

NR06. Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 271–288. Springer, Heidelberg, May / June 2006.

Pei10. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Heidelberg, August 2010.

PSDS17. Thomas Plantard, Arnaud Sipasseuth, Cedric Dumondelle, and Willy Susilo. DRS. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

Pre23. Thomas Prest. A key-recovery attack against mitaka in the t-probing model. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 205–220. Springer, Heidelberg, May 2023.

PFH+22. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

SKLN22. Eun-Young Seo, Young-Sik Kim, Joon-Woo Lee, and Jong-Seon No. Peregrine: Submission to the Korea post-quantum cryptography competition round 1. Technical report, 2022. available at https://www.kpqc.or.kr/competition.html.

TW20. Mehdi Tibouchi and Alexandre Wallet. One bit is all it takes: a devastating timing attack on BLISS's non-constant time sign flips. *Journal of Mathematical Cryptology*, 15(1):131–142, 2020.

YD18. Yang Yu and Léo Ducas. Learning strikes again: The case of the DRS signature scheme. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 525–543. Springer, Heidelberg, December 2018.