Cryptanalysis of the Peregrine Lattice-Based Signature Scheme

PKC 2024

Xiuhan Lin, Moeto Suzuki, Shiduo Zhang, Thomas Espitau, Yang Yu, Mehdi Tibouchi, Masayuki Abe

山东大学 SHANDONG UNIVERSITY 京都大学 KYOTO UNIVERSITY 清華大学 Tsinghua University PQSHIELD NTT

The Cryptanalysis of Peregrine

- Target: Peregrine[1]
    - the first round of the Korean PQC competition candidate in 2023

[1] https://www.kpqc.or.kr/competition.html.

The Cryptanalysis of Peregrine

- Target: Peregrine[1]
  - the first round of the Korean PQC competition candidate in 2023
- Technique: "parallelepiped-learning" +"lattice decoding"
  - parallelepiped-learning ⇒ the approximate key found
  - lattice decoding ⇒ fully recovers the secret from the approximations

[1] https://www.kpqc.or.kr/competition.html.

The Cryptanalysis of Peregrine

- Target: Peregrine[1]
  - the first round of the Korean PQC competition candidate in 2023
- Technique: “parallelepiped-learning” +“lattice decoding”
  - parallelepiped-learning $\Rightarrow$ the approximate key found
  - lattice decoding $\Rightarrow$ fully recovers the secret from the approximations
- Cost: the signature samples required for practical attacks
  - $\approx 25,000$ for the reference implementation
  - ≈ 11 million for the specification version

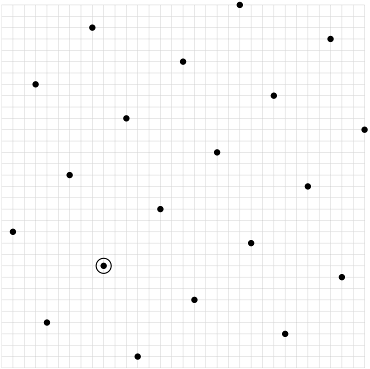
[1] https://www.kpqc.or.kr/competition.html.

Outline

- Background
- The Peregrine signature scheme
- Learning a hidden transformation
- Practical key recovery attack

Background

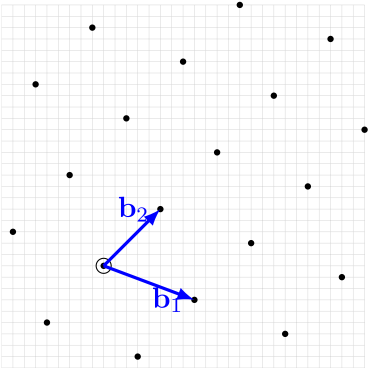
Lattice



Lattice

A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$.

Lattice

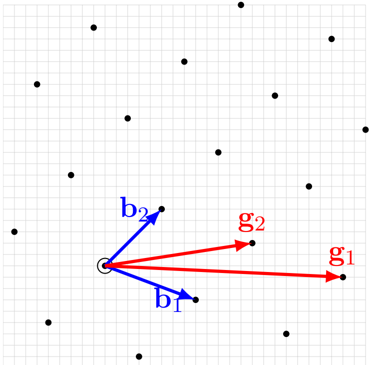


Lattice

A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$.

A lattice is generated by its basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n) \in \mathbb{R}^{m \times n}$, i.e. $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$.

Lattice



Lattice

A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$.

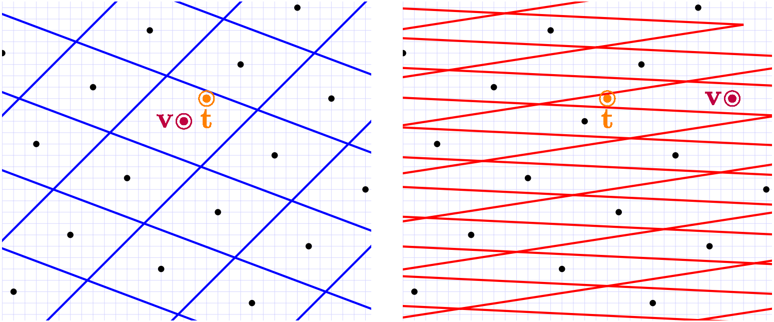
A lattice is generated by its basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n) \in \mathbb{R}^{m \times n}$, i.e. $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$.

$\mathcal{L}$ has infinitely many bases

$\mathbf{B}$ is good, $\mathbf{G}$ is bad.

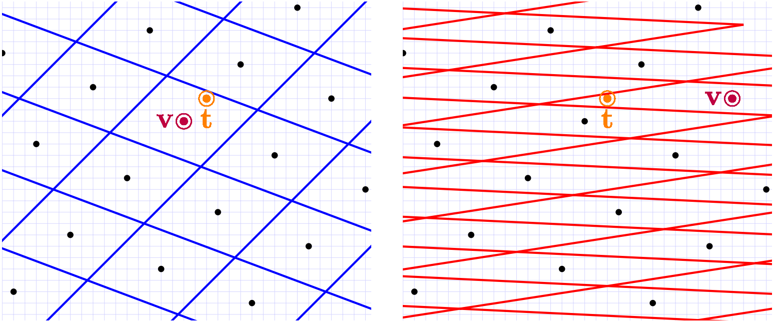
Parallelepiped

Each basis defines a parallelepiped $\mathcal{P}(\mathbf{B}) = \left\{\mathbf{xB} \mid \mathbf{x} \in \left[-\frac{1}{2}, \frac{1}{2}\right)^n\right\}$.



Parallelepiped

Each basis defines a parallelepiped $\mathcal{P}(\mathbf{B}) = \left\{ \mathbf{xB} \mid \mathbf{x} \in \left[-\frac{1}{2}, \frac{1}{2}\right)^n \right\}$.



Babai's round-off algorithm outputs $\mathbf{v} \in \mathcal{L}$ such that $\mathbf{v} - \mathbf{t} \in \mathcal{P}$.

Hash-and-sign construction

Hash-and-sign

- signing: to solve the approximate closest vector problem (CVP)
- evolution: GGH, NTRUSign $\rightarrow$ GPV $\rightarrow$ Falcon, Mitaka

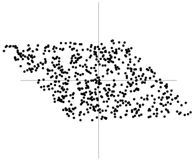
Hash-and-sign construction

Hash-and-sign

- signing: to solve the approximate closest vector problem (CVP)
- evolution: GGH, NTRUSign $\rightarrow$ GPV $\rightarrow$ Falcon, Mitaka

GGH, NTRUSign use deterministic round-off algorithm to solve the CVP.

- $\mathbf{v} - \mathbf{t} \in \mathcal{P}(\mathbf{B})$, the distribution of signatures leaks information of $\mathbf{B}$
- broken by parallelepiped-learning attacks [NR06][2]



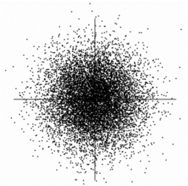
Parallelepiped. Insecure!

[2] [NR06]: Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. Nguyen and Regev.

GPV framework

[GPV08][3] presented a provably secure framework.

- deterministic round-off algorithm $\Rightarrow$ trapdoor sampler
- randomizing the rounding with random Gaussian sampling on lattice
- the distribution of signatures is independent of the secret



Gaussian

[3][GPV08]: Trapdoors for Hard Lattices and New Cryptographic Constructions. Gentry, Peikert, Vaikuntanathan.

Falcon

Falcon signature scheme[4]

- selected by NIST for standardization in 2022
- initiated with GPV framework over NTRU lattices
- **advantages:** low bandwidth, good efficiency
- **disadvantages:** complicated, due to Gaussian sampling and floating-point operations

[4] https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

Falcon

Falcon signature scheme[4]

- selected by NIST for standardization in 2022
- initiated with GPV framework over NTRU lattices
- **advantages:** low bandwidth, good efficiency
- **disadvantages:** complicated, due to Gaussian sampling and floating-point operations

Designing a simpler and comparably efficient variant of Falcon is a tempting choice!

[4] https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

The Peregrine signature scheme

Peregrine

Peregrine signature scheme

- one of candidates in the 1st round of the KPQC competition
- the high speed version of Falcon
- Gaussian sampling ✖, centered binomial distribution ✔
- simpler, along with comparable efficiency, easy to mask

Peregrine does not offer a proof of security!!!

The procedure of signing

The signing of Peregrine is in essence the randomized version of Babai's round-off algorithm.

- by adding a binomial vector (J_1, J_2), instead of using Gaussian distribution

Signing

Input: NTRU trapdoor basis $\mathbf{B}$, center $\mathbf{c}$.

Output: random lattice point $\mathbf{s} \in \mathcal{L}(\mathbf{B}) - \mathbf{c}$.

1: $(J_1, J_2) \leftarrow (B_{\mu_1}^{n/2}, B_{\mu_2}^{n/2})$

2: $\mathbf{z} = \lfloor \mathbf{B}^{-1}\mathbf{c} \rceil + (J_1, J_2)$

3: $\mathbf{v} = \mathbf{B}\mathbf{z}$

4: $\mathbf{s} = \mathbf{v} - \mathbf{c}$

5: **return** $\mathbf{s}$

The centered binomial distribution B_μ is defined over $[-\frac{\mu}{2}, \frac{\mu}{2}] \cap \mathbb{Z}$.

Signature distribution

Practical distribution

We have $\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \mathbf{B}_{f,g} \cdot \begin{pmatrix} R_1 - J_1 \\ R_2 - J_2 \end{pmatrix}$ where $(R_1, R_2) \sim U([-1/2, 1/2)^n)$.

Signature distribution

Practical distribution

We have $\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \mathbf{B}_{f,g} \cdot \begin{pmatrix} R_1 - J_1 \\ R_2 - J_2 \end{pmatrix}$ where $(R_1, R_2) \sim U([-1/2, 1/2)^n)$.

- the distribution of (s_1, s_2) is a hidden linear transformation (i.e. $\mathbf{B}_{f,g}$) of a known distribution

Signature distribution

Practical distribution

We have $\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \mathbf{B}_{f,g} \cdot \begin{pmatrix} R_1 - J_1 \\ R_2 - J_2 \end{pmatrix}$ where $(R_1, R_2) \sim U([-1/2, 1/2)^n)$.

- the distribution of (s_1, s_2) is a hidden linear transformation (i.e. $\mathbf{B}_{f,g}$) of a known distribution
- we perform practical key recovery attacks against Peregrine by learning the hidden linear transformation

Secret key leakage

The Peregrine signatures are always in adjacent parallelepipeds, rather than a sole parallelepiped.

Adjacent parallelepipeds ✔

Sole parallelepiped ✖

Secret key leakage

The Peregrine signatures are always in adjacent parallelepipeds, rather than a sole parallelepiped.

Adjacent parallelepipeds ✔

Sole parallelepiped ✖

Peregrine are also insecure!!!

Secret key leakage

The Peregrine signatures are always in adjacent parallelepipeds, rather than a sole parallelepiped.

Adjacent parallelepipeds ✔

Sole parallelepiped ✖

Peregrine are also insecure!!!

- the distribution of signatures would leak information of the secret key
- learn the hidden transformation by parallelepiped-learning of [NR06]

Concrete parameters

In this work, we focus on the parameter set of Peregrine–512.

Concrete parameters

In this work, we focus on the parameter set of Peregrine–512.

There are some discrepancies between the reference implementation and the official specification of Peregrine.

Concrete parameters

In this work, we focus on the parameter set of Peregrine–512.

There are some discrepancies between the reference implementation and the official specification of Peregrine.

- **key generation:**
  - in the specification, the coefficients of (f, g) are drawn from B_{26}, and it checks if the Gram–Schmidt norms of $\mathbf{B}_{f,g}$ are less than $1.17\sqrt{q}$
  - in the reference implementation, this check is commented out

Concrete parameters

In this work, we focus on the parameter set of Peregrine–512.

There are some discrepancies between the reference implementation and the official specification of Peregrine.

- **key generation:**
    - in the specification, the coefficients of (f, g) are drawn from B_{26}, and it checks if the Gram–Schmidt norms of $\mathbf{B}_{f,g}$ are less than $1.17\sqrt{q}$
    - in the reference implementation, this check is commented out
- **the signing:**
    - the specification suggests $\mu_1 = \mu_2 = 26$
    - the reference implementation in effect use $(\mu_1, \mu_2) = (6, 0)$

Learning a hidden transformation

Parallelepiped-learning of [NR06] revisit

Definition 1 (The Hidden Parallelepiped Problem)

Given $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathrm{GL}_n(\mathbb{R})$ and a certain number of independent parallelepiped samples $\mathbf{y} = \mathbf{B}\mathbf{x}$ with $\mathbf{x} \leftarrow U([-1, 1])$, find an approximation of $\pm\mathbf{b}_i$'s.

Parallelepiped-learning of [NR06] revisit

Definition 1 (The Hidden Parallelepiped Problem)

Given $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathrm{GL}_n(\mathbb{R})$ and a certain number of independent parallelepiped samples $\mathbf{y} = \mathbf{Bx}$ with $\mathbf{x} \leftarrow U([-1, 1])$, find an approximation of $\pm\mathbf{b}_i$'s.

Solving the Hidden Parallelepiped Problem

- the covariance leakage: $\mathbf{K} = \mathbf{B} \cdot \mathbf{Cov}[\mathbf{xx}^t] \cdot \mathbf{B}^t = \mathbf{BB}^t/3$
- the approximate Gram matrix: $\mathbf{K} = 3\mathbf{K} = \mathbf{BB}^t$
- compute factor $\mathbf{L} = \mathbf{P}^t$ such that $\mathbf{K}^{-1} = \mathbf{PP}^t$
- by multiplying $\mathbf{L}$, $\mathbf{C} = \mathbf{LB}$ is orthogonal
- the local minima $\pm\mathbf{c}_i$ can be solved by gradient descent
- by multiplying $\mathbf{L}^{-1}$, the approximation of $\pm\mathbf{b}_i$ found

Hidden Transformation Problem

The Nguyen-Regev parallelepiped-learning attack [NR06] can be extended to more general *Hidden Transformation Problem* (HTP).

Hidden Transformation Problem

The Nguyen-Regev parallelepiped-learning attack [NR06] can be extended to more general *Hidden Transformation Problem* (HTP).

Definition 2 (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.

Hidden Transformation Problem

The Nguyen-Regev parallelepiped-learning attack [NR06] can be extended to more general *Hidden Transformation Problem* (HTP).

Definition 2 (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.

For Peregrine,

$$D_i = \begin{cases} U([-1/2, 1/2)) + B_{\mu_1} & \text{for } 1 \leq i \leq n/2; \\ U([-1/2, 1/2)) + B_{\mu_2} & \text{for } n/2+1 \leq i \leq n. \end{cases}$$

.

Key recovery algorithm

Our key recovery algorithm

1. distribution deformation
2. gradient descent

Step 1: Distribution deformation

The covariance leakage

- $\mathbf{Cov}[D(\mathbf{B})] = \mathbf{B} \cdot \mathbf{Cov}[D] \cdot \mathbf{B}^t$
- helps to reduce the general HTP to the case in which the covariance leakage is $\mathbf{I}_n$

Step 1: Distribution deformation

The covariance leakage

- $\mathbf{Cov}[D(\mathbf{B})] = \mathbf{B} \cdot \mathbf{Cov}[D] \cdot \mathbf{B}^t$
- helps to reduce the general HTP to the case in which the covariance leakage is $\mathbf{I}_n$

The procedure of distribution deformation

- the covariance leakage $\mathbf{K} = \mathbf{Cov}[D(\mathbf{B})]$
- compute $\mathbf{L} = \mathbf{P}^t$ such that $\mathbf{P}\mathbf{P}^t = \mathbf{K}^{-1}$
- $\mathbf{C} = \mathbf{LB}$ such that $\mathbf{Cov}[D(\mathbf{C})] = \mathbf{I}_n$
- $\mathbf{C}$ is orthogonal when $\mathbf{Cov}[D] = \mathbf{I}_n$

Step 1: Distribution deformation

The covariance leakage

- $\mathbf{Cov}[D(\mathbf{B})] = \mathbf{B} \cdot \mathbf{Cov}[D] \cdot \mathbf{B}^t$
- helps to reduce the general HTP to the case in which the covariance leakage is $\mathbf{I}_n$

The procedure of distribution deformation

- the covariance leakage $\mathbf{K} = \mathbf{Cov}[D(\mathbf{B})]$
- compute $\mathbf{L} = \mathbf{P}^t$ such that $\mathbf{P}\mathbf{P}^t = \mathbf{K}^{-1}$
- $\mathbf{C} = \mathbf{L}\mathbf{B}$ such that $\mathbf{Cov}[D(\mathbf{C})] = \mathbf{I}_n$
- $\mathbf{C}$ is orthogonal when $\mathbf{Cov}[D] = \mathbf{I}_n$

Distribution deformation reduces the HTP instance regarding $(D, \mathbf{B})$ to the one regarding $(D, \mathbf{C})$ such that $\mathbf{Cov}[D(\mathbf{C})] = \mathbf{I}_n$ and $\mathbf{Cov}[D] = \mathbf{I}_n$.

Step 2: Gradient descent

Let $\alpha_i = \mathbb{E}[z_i^4]$. The fourth moment of $D(\mathbf{C})$ and its gradient:

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

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

Step 2: Gradient descent

Let $\alpha_i = \mathbb{E}[z_i^4]$. The fourth moment of $D(\mathbf{C})$ and its gradient:

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

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

Lemma 1

Suppose that $\alpha_i < 3$ for all $1 \le i \le n$, 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.

Step 2: Gradient descent

Let $\alpha_i = \mathbb{E}[z_i^4]$. The fourth moment of $D(\mathbf{C})$ and its gradient:

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

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

Lemma 1

Suppose that $\alpha_i < 3$ for all $1 \le i \le n$, 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.

Therefore, the local minima $\mathbf{c}_i$ can be solved by gradient descent[TW20][5].

[5] [TW20]: One bit is all it takes: a devastating timing attack on BLISS's non-constant time sign flips. Tibouchi and Wallet.

The case of [NR06]

In [NR06], $D = U([-1, 1])$, the fourth moment function:

$$M_{D(\mathbf{C}),4}(\mathbf{w}) = \frac{1}{3}\|\mathbf{w}\|^4 - \frac{2}{15}\sum_{i=1}^{n}\langle \mathbf{c}_i, \mathbf{w}\rangle^4,$$

and its gradient:

$$\nabla M_{D(\mathbf{C}),4}(\mathbf{w}) = \frac{4}{3}\mathbf{w} - \frac{8}{15}\sum_{i=1}^{n}\langle \mathbf{c}_i, \mathbf{w}\rangle^3 \mathbf{c}_i.$$

The case of Peregrine

For specification version, $\mu_1 = \mu_2 = 26$:

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

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

For reference implementation version, $(\mu_1, \mu_2) = (6, 0)$:

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

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

Practical key recovery attacks

Lattice decoding of [Pre23]

Let $\mathbf{b} = (b^{(1)}, b^{(2)}) \in \mathcal{L}_{\mathsf{NTRU}}$ be the secret vector and $\mathbf{b}' = \left((b')^{(1)}, (b')^{(2)}\right)$ be the approximation of $\mathbf{b}$.

Lattice decoding of [Pre23]

Let $\mathbf{b} = (b^{(1)}, b^{(2)}) \in \mathcal{L}_{\mathsf{NTRU}}$ be the secret vector and $\mathbf{b}' = ((b')^{(1)}, (b')^{(2)})$ be the approximation of $\mathbf{b}$.

Prest's decoding technique [Pre23][6]

- Selecting a certain threshold $\varepsilon \in (0, 1/2)$
  - For $\mathbf{e} = \mathbf{b}' - \mathbf{b}$, at least half of the coefficients of $\mathbf{e}$ are in $[-\varepsilon, \varepsilon]$
  - No coefficients of $\mathbf{e}$ in absolute norm exceeds $1 - \varepsilon$

[6] [Pre23]: A key-recovery attack against mitaka in the t-probing model. Prest.

Lattice decoding of [Pre23]

Let $\mathbf{b} = (b^{(1)}, b^{(2)}) \in \mathcal{L}_{\mathsf{NTRU}}$ be the secret vector and $\mathbf{b}' = ((b')^{(1)}, (b')^{(2)})$ be the approximation of $\mathbf{b}$.

Prest's decoding technique [Pre23][6]

- Selecting a certain threshold $\varepsilon \in (0, 1/2)$
  - For $\mathbf{e} = \mathbf{b}' - \mathbf{b}$, at least half of the coefficients of $\mathbf{e}$ are in $[-\varepsilon, \varepsilon]$
  - No coefficients of $\mathbf{e}$ in absolute norm exceeds $1 - \varepsilon$
- The difference $\mathbf{d} = \lfloor \mathbf{b}' \rceil - \mathbf{b} = (d^{(1)}, d^{(2)})$
  - zeros in at least $n/2$ coefficients

[6] [Pre23]: A key-recovery attack against mitaka in the t-probing model. Prest.

Lattice decoding of [Pre23]

Let $\mathbf{b} = (b^{(1)}, b^{(2)}) \in \mathcal{L}_{\mathsf{NTRU}}$ be the secret vector and $\mathbf{b}' = ((b')^{(1)}, (b')^{(2)})$ be the approximation of $\mathbf{b}$.

Prest's decoding technique [Pre23][6]

- Selecting a certain threshold $\varepsilon \in (0, 1/2)$
  - For $\mathbf{e} = \mathbf{b}' - \mathbf{b}$, at least half of the coefficients of $\mathbf{e}$ are in $[-\varepsilon, \varepsilon]$
  - No coefficients of $\mathbf{e}$ in absolute norm exceeds $1 - \varepsilon$
- The difference $\mathbf{d} = \lfloor \mathbf{b}' \rceil - \mathbf{b} = (d^{(1)}, d^{(2)})$
  - zeros in at least $n/2$ coefficients
  - for NTRU equation, $b^{(1)} + b^{(2)} \cdot h = 0 \bmod q$, then

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

[6] [Pre23]: A key-recovery attack against mitaka in the t-probing model. Prest.

Lattice decoding of [Pre23]

Let $\mathbf{b} = (b^{(1)}, b^{(2)}) \in \mathcal{L}_{\mathsf{NTRU}}$ be the secret vector and $\mathbf{b}' = ((b')^{(1)}, (b')^{(2)})$ be the approximation of $\mathbf{b}$.

Prest's decoding technique [Pre23][6]

- Selecting a certain threshold $\varepsilon \in (0, 1/2)$
  - For $\mathbf{e} = \mathbf{b}' - \mathbf{b}$, at least half of the coefficients of $\mathbf{e}$ are in $[-\varepsilon, \varepsilon]$
  - No coefficients of $\mathbf{e}$ in absolute norm exceeds $1 - \varepsilon$
- The difference $\mathbf{d} = \lfloor \mathbf{b}' \rceil - \mathbf{b} = (d^{(1)}, d^{(2)})$
  - zeros in at least $n/2$ coefficients
  - for NTRU equation, $b^{(1)} + b^{(2)} \cdot h = 0 \bmod q$, then

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

The secret $\mathbf{b}$ can be fully recovered by solving linear system for $\mathbf{d}$.

[6] [Pre23]: A key-recovery attack against mitaka in the t-probing model. Prest.

Probability-based guessing strategy

Our approach is slightly different from the trick of [Pre23].

Probability-based guessing strategy

Our approach is slightly different from the trick of [Pre23].

- Without exploiting a threshold ε
- Selecting $n/2$ coefficients which are correctly rounded with the highest probability

Probability-based guessing strategy

Our approach is slightly different from the trick of [Pre23].

- Without exploiting a threshold ε
- Selecting $n/2$ coefficients which are correctly rounded with the highest probability

Lemma 2

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:

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

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

Probability-based guessing strategy

Our approach is slightly different from the trick of [Pre23].

- Without exploiting a threshold ε
- Selecting $n/2$ coefficients which are correctly rounded with the highest probability

Lemma 2

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:

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

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

The standard deviation is inversely proportional to required signatures N: $\sigma \approx C_\sigma/\sqrt{N}$ and constant C_σ can be derived by curve fitting.

Experimental results

For reference implementation

- signature samples: $\approx 25,000$
- running time: < 0.5 hours

$N \times 10^{-3}$	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

Experimental results

For the specification version

- signature samples: ≈ 11 million
- running time: < 20 hours

$N \times 10^{-6}$	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

Conclusion

Conclusion

We present practical key recovery attacks against Peregrine.

- we can practically break two versions of Peregrine-512 by using a relatively small number of signatures in a few hours
- The same attack can be extended to the case of Peregrine-1024

Conclusion

We present practical key recovery attacks against Peregrine.

- we can practically break two versions of Peregrine-512 by using a relatively small number of signatures in a few hours
- The same attack can be extended to the case of Peregrine-1024

More efficient countermeasures against statistical attacks need further investigations!

Thank you!