Message Recovery in NTRU based on CVP

M.Adamoudis, *K.A. Draziotis* and E. Poimenidou

NuTMiC 2024, June 24 – June 26, 2024, Szczecin, Poland

Aristotle Uni. of Thessaloniki, School of Informatics & CERN.
The second author was co funded by SECUR-EU. The SECUR-EU project funded under Grant Agreement 101128029 is supported by the European Cybersecurity Competence Centre.

History of NTRU cryptosystem

NTRU encrypt is a public-key encryption scheme developed by Jeff Hoffstein, Jill Pipher, and Joseph H. Silverman in 1996. Over time, this cryptographic system has evolved, leading to several variants designed to enhance security and efficiency.

History of NTRU cryptosystem

NTRU encrypt is a public-key encryption scheme developed by Jeff Hoffstein, Jill Pipher, and Joseph H. Silverman in 1996. Over time, this cryptographic system has evolved, leading to several variants designed to enhance security and efficiency.

These developments include NTRU-HPS (Hoffstein-Pipher-Silverman), NTRU-Prime, and NTRU-HRSS.

History of NTRU cryptosystem

NTRU encrypt is a public-key encryption scheme developed by Jeff Hoffstein, Jill Pipher, and Joseph H. Silverman in 1996. Over time, this cryptographic system has evolved, leading to several variants designed to enhance security and efficiency.

These developments include NTRU-HPS (Hoffstein-Pipher-Silverman), NTRU-Prime, and NTRU-HRSS.

We believe that NTRU based schemes are post quantum secure.

- It was implemented in openssh ver.9.0 (hybrid Streamlined NTRU Prime + x25519 key exchange method)[1]

[1] https://www.openssh.com/txt/release-9.0
[2] https://www.wolfssl.com/wolfssl-with-ntru-cipher-suites/
[3] https://github.com/google/boringssl
[4] https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms

- It was implemented in openssh ver.9.0 (hybrid Streamlined NTRU Prime + x25519 key exchange method)[1]
- wolfssl+ [2]

[1] https://www.openssh.com/txt/release-9.0
[2] https://www.wolfssl.com/wolfssl-with-ntru-cipher-suites/
[3] https://github.com/google/boringssl
[4] https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms

- It was implemented in openssh ver.9.0 (hybrid Streamlined NTRU Prime + x25519 key exchange method)[1]
- wolfssl+ [2]
- boringSSL of Google[3]

[1] https://www.openssh.com/txt/release-9.0
[2] https://www.wolfssl.com/wolfssl-with-ntru-cipher-suites/
[3] https://github.com/google/boringssl
[4] https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms

- It was implemented in openssh ver.9.0 (hybrid Streamlined NTRU Prime + x25519 key exchange method)[1]
- wolfssl+ [2]
- boringSSL of Google[3]
- In Application Layer Transport Security (ALTS) of Google, they use NTRU-HRSS in hybrid set up [4].

[1] https://www.openssh.com/txt/release-9.0
[2] https://www.wolfssl.com/wolfssl-with-ntru-cipher-suites/
[3] https://github.com/google/boringssl
[4] https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms

Our Goal

We present a message recovery attack applicable to all NTRU variants, assuming the knowledge of 2 bits of each coefficient of a polynomial which is a multiple of the nonce.

Our Goal

- Such assumptions are commonly used in the cryptanalysis of many cryptographic primitives, such as (EC)DSA[5], where if we know some bits of many ephemeral keys we can compute the secret key,

[5] M. Adamoudis, K. A. Draziotis and D. Poulakis, Enhancing a DSA attack, CAI 2019, p. 13-25. LNCS 11545, Springer 2019.

[6] Alexander May and Julian Nowakowski, Too Many Hints ? When LLL Breaks LWE, 2023,`https://eprint.iacr.org/2023/777.pdf`

Our Goal

- Such assumptions are commonly used in the cryptanalysis of many cryptographic primitives, such as (EC)DSA[5], where if we know some bits of many ephemeral keys we can compute the secret key,
- in RSA (in Coppersmith like attacks), where if we know some bits of the unknown prime numbers we can compute the prime numbers of the modulus RSA,

[5] M. Adamoudis, K. A. Draziotis and D. Poulakis, Enhancing a DSA attack, CAI 2019, p. 13-25. LNCS 11545, Springer 2019.

[6] Alexander May and Julian Nowakowski, Too Many Hints ? When LLL Breaks LWE, 2023,https://eprint.iacr.org/2023/777.pdf

Our Goal

- Such assumptions are commonly used in the cryptanalysis of many cryptographic primitives, such as (EC)DSA[5], where if we know some bits of many ephemeral keys we can compute the secret key,
- in RSA (in Coppersmith like attacks), where if we know some bits of the unknown prime numbers we can compute the prime numbers of the modulus RSA,
- and more recently an attack to kyber[6] where if we know some information about the LWE secret through hints, modeled as inner products with known vectors, we compute the secret key.

[5] M. Adamoudis, K. A. Draziotis and D. Poulakis, Enhancing a DSA attack, CAI 2019, p. 13-25. LNCS 11545, Springer 2019.

[6] Alexander May and Julian Nowakowski, Too Many Hints ? When LLL Breaks LWE, 2023,`https://eprint.iacr.org/2023/777.pdf`

Introduction to NTRU

In all the NTRU variants, we have a parameter N (current values are > 500), which is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^{\ell}$, current values of $\ell > 10$).

Introduction to NTRU

In all the NTRU variants, we have a parameter N (current values are > 500), which is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^{\ell}$, current values of $\ell > 10$).

The four sample spaces $\mathcal{M}_z$, for $z \in \{f, g, m, r\}$, where $(f(x), g(x))$ is the secret key, $m(x)$ is the message and $r(x)$ the nonce (or the ephemeral key).

Introduction to NTRU

In all the NTRU variants, we have a parameter N (current values are > 500), which is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^{\ell}$, current values of $\ell > 10$).

The four sample spaces $\mathcal{M}_z$, for $z \in \{f, g, m, r\}$, where $(f(x), g(x))$ is the secret key, $m(x)$ is the message and $r(x)$ the nonce (or the ephemeral key).

Usually all the sample spaces are subset of the set of ternary polynomials of degree N i.e. polynomials having only $1, 0, -1$ as coefficients.

Introduction to NTRU

- In all the NTRU variants, we have a parameter N (current values are > 500), which is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^{\ell}$, current values of $\ell > 10$).
- The four sample spaces $\mathcal{M}_z$, for $z \in \{f, g, m, r\}$, where $(f(x), g(x))$ is the secret key, $m(x)$ is the message and $r(x)$ the nonce (or the ephemeral key).
- Usually all the sample spaces are subset of the set of ternary polynomials of degree N i.e. polynomials having only $1, 0, -1$ as coefficients.
- We also set, the polynomial ring $\mathcal{R} = \mathbb{Z}[x]/\langle D(x)\rangle$, $\deg D(x) = N$ and $\star$ is the multiplication in the ring $\mathcal{R}$.

Introduction to NTRU

The secret key is $(f(x), g(x)) \xleftarrow{\$} \mathcal{M}_f \times \mathcal{M}_g$.

Introduction to NTRU

The secret key is $(f(x), g(x)) \xleftarrow{\$} \mathcal{M}_f \times \mathcal{M}_g$.

The public key $h(x) = 3f^{-1}(x) \star g(x) \pmod{q, D(x)}$.

Introduction to NTRU

The secret key is $(f(x), g(x)) \xleftarrow{\$} \mathcal{M}_f \times \mathcal{M}_g$.

The public key $h(x) = 3f^{-1}(x) \star g(x) \pmod{q, D(x)}$.

We follow here NTRU-HPS, so $D(x) = x^N - 1$.

Introduction to NTRU

The secret key is $(f(x), g(x)) \xleftarrow{\$} \mathcal{M}_f \times \mathcal{M}_g$.

The public key $h(x) = 3f^{-1}(x) \star g(x) \pmod{q, D(x)}$.

We follow here NTRU-HPS, so $D(x) = x^N - 1$.

We also set $\Phi_N(x) = x^{N-1} + \cdots + x + 1$

NTRU problem

The problem of distinguishing $h(x)$ from uniform elements in $\mathcal{R}/q$ is called *decision NTRU problem.*

NTRU problem

The problem of distinguishing $h(x)$ from uniform elements in $\mathcal{R}/q$ is called *decision NTRU problem.*

While, the problem of finding the private key $(f(x), g(x))$, given $h(x)$, is referred to as the *search NTRU problem*.

Introduction to NTRU

The (secret) vector $(f, 3g)$ belongs to the lattice

$$\mathcal{L}_{NTRU} = \{(a(x), b(x)) \in \mathcal{R}^2 : b(x) = a(x) \star h(x) \pmod{q}\},$$

Introduction to NTRU

The (secret) vector $(f, 3g)$ belongs to the lattice

$$\mathcal{L}_{NTRU} = \{(a(x), b(x)) \in \mathcal{R}^2 : b(x) = a(x) \star h(x) \pmod{q}\},$$

A basis is given from the rows of the matrix

$$M_{\mathbf{h}} = \left[\begin{array}{c|c} I_N & \mathbf{C}(\mathbf{h}) \\ \hline \mathbf{0}_N & qI_N \end{array}\right].$$

where, the upper right block $\mathbf{C}(\mathbf{h})$, is the cyclic matrix generated by the vector $\mathbf{h} = (h_0, ..., h_{N-1})$, where
$h(x) = h_{N-1}x^{N-1} + \cdots + h_1 x + h_0$.

Encrypt-Decrypt

To encrypt a message $m(x) \in \mathcal{M}_m$

we choose a random ephemeral key $r(x) \in \mathcal{M}_r$ and we compute the ciphertext,

$$c(x) \leftarrow h(x) \star r(x) + m(x) \mod q.$$

Encrypt-Decrypt

To encrypt a message $m(x) \in \mathcal{M}_m$

we choose a random ephemeral key $r(x) \in \mathcal{M}_r$ and we compute the ciphertext,

$$c(x) \leftarrow h(x) \star r(x) + m(x) \mod q.$$

To decrypt, first we set $a(x) \leftarrow c(x) \star f(x) \mod (q, D(x))$

Encrypt-Decrypt

To encrypt a message $m(x) \in \mathcal{M}_m$

we choose a random ephemeral key $r(x) \in \mathcal{M}_r$ and we compute the ciphertext,

$$c(x) \leftarrow h(x) \star r(x) + m(x) \mod q.$$

To decrypt, first we set $a(x) \leftarrow c(x) \star f(x) \mod (q, D(x))$

Then, $m(x) \leftarrow \mathrm{centerlift}\Big(a(x) \star f_3(x) \mod \big(3, \Phi_N(x)\big)\Big)$

Main idea of the attack

We multiply the encryption equation by an integer k (we shall choose it later), so we get

$$km(x) = kc(x) - kh(x) \star r(x) = b_k(x) - U_k(x) \pmod{q, D(x)}.$$

Main idea of the attack

We multiply the encryption equation by an integer k (we shall choose it later), so we get

$$km(x) = kc(x) - kh(x) \star r(x) = b_k(x) - U_k(x) \pmod{q, D(x)}.$$

Note that knowing $U_k(x) = kh(x) \star r(x)$ is equivalent to knowing $m(x)$.

Main idea of the attack

We work with the lattice $\mathcal{L}_k$ generated by the rows of

$$M_k = \left[\begin{array}{c|c} I_N & -kI_N \\ \hline \mathbf{0}_N & qI_N \end{array}\right]$$

Main idea of the attack

We work with the lattice $\mathcal{L}_k$ generated by the rows of

$$M_k = \left[\begin{array}{c|c} I_N & -kI_N \\ \hline \mathbf{0}_N & qI_N \end{array}\right]$$

Note that this lattice is independent from the public key.

Main idea of the attack

We work with the lattice $\mathcal{L}_k$ generated by the rows of

$$M_k = \left[\begin{array}{c|c} I_N & -kI_N \\ \hline \mathbf{0}_N & qI_N \end{array} \right]$$

Note that this lattice is independent from the public key.

k, is not a random integer, and we shall choose it later.

The attack

For the previous Lattice under some plausible conditions we can compute λ_1.

The attack

For the previous Lattice under some plausible conditions we can compute λ_1.

Proposition. Let k, N and q be positive integers with $q \geq (k+1)\sqrt{k^2+1}$. We set

$$M_k = \left[\begin{array}{c|c} I_N & -kI_N \\ \hline \mathbf{0}_N & qI_N \end{array}\right].$$

Let $\mathcal{L}_k$ be the lattice generated by the rows of M_k. Then, $\lambda_1(\mathcal{L}_k) = \sqrt{k^2+1}$.

the proof

It is enough to prove that for all non-zero $\mathbf{v} \in \mathcal{L}_k$ we have $\|\mathbf{v}\| \geq \sqrt{k^2+1}$. Since the first row of M_k has length $\sqrt{k^2+1}$ we are done.

the proof

It is enough to prove that for all non-zero $\mathbf{v} \in \mathcal{L}_k$ we have $\|\mathbf{v}\| \geq \sqrt{k^2+1}$. Since the first row of M_k has length $\sqrt{k^2+1}$ we are done.

Suppose that there is a vector $\mathbf{v} \in \mathcal{L}_k \setminus \{\mathbf{0}\}$ such that

$$\|\mathbf{v}\| < \sqrt{k^2+1}. \tag{1}$$

the proof

It is enough to prove that for all non-zero $\mathbf{v} \in \mathcal{L}_k$ we have $\|\mathbf{v}\| \geq \sqrt{k^2+1}$. Since the first row of M_k has length $\sqrt{k^2+1}$ we are done.

Suppose that there is a vector $\mathbf{v} \in \mathcal{L}_k \setminus \{\mathbf{0}\}$ such that

$$\|\mathbf{v}\| < \sqrt{k^2+1}. \tag{1}$$

Let $\mathbf{b}_1, \ldots, \mathbf{b}_{2N}$ be the rows of the matrix M_k. Since $\mathbf{v} \in \mathcal{L}_k$, there are integers $l_1, \ldots, l_{2N}$ such that,

$$\mathbf{v} = l_1\mathbf{b}_1 + \cdots + l_{2N}\mathbf{b}_{2N} =$$

$$(l_1, \ldots, l_N, -l_1k + ql_{N+1}, \ldots, -l_Nk + ql_{2N})$$

From the inequality (1) we get

the proof

$$\begin{cases} |l_1|, |l_2|, \dots, |l_N| < \sqrt{k^2+1} \\ |-l_1 k + q l_{N+1}| < \sqrt{k^2+1} \\ \dots \\ |-l_N k + q l_{2N}| < \sqrt{k^2+1} \end{cases} \tag{2}$$

So we can easily see that for $i = 1, \dots, N$ we get

$$|l_i k| < \sqrt{k^2+1}\,k. \tag{3}$$

the proof

Case 1: not all the integers $l_{N+1}, l_{N+2}, \ldots, l_{2N}$ are zero. Without loss of generality, say l_{N+j} is not zero for some $j \in \{1, \ldots, N\}$. Then from (3) and (2), we get

$$\|\mathbf{v}\| \geq |-l_j k + q l_{N+j}| \geq |l_{N+j}|q - |l_j k| > q - \sqrt{k^2+1}k \geq \sqrt{k^2+1},$$

which contradicts to inequality (1).

the proof

Case 1: not all the integers $l_{N+1}, l_{N+2}, \dots, l_{2N}$ are zero. Without loss of generality, say l_{N+j} is not zero for some $j \in \{1, \dots, N\}$. Then from (3) and (2), we get

$$\|\mathbf{v}\| \geq |-l_j k + q l_{N+j}| \geq |l_{N+j}|q - |l_j k| > q - \sqrt{k^2+1}k \geq \sqrt{k^2+1},$$

which contradicts to inequality (1).

Case 2: Let $l_{N+1} = l_{N+2} = \cdots = l_{2N} = 0$.

In this case

$$\mathbf{v} = (l_1, \dots, l_N, -l_1 k, \dots, -l_N k).$$

Then,

$$\|\mathbf{v}\| = \sqrt{l_1^2(1+k^2) + l_2^2(1+k^2) + \cdots + l_N^2(1+k^2)} > \sqrt{k^2+1},$$

which contradicts our hypothesis (1).

The attack

Say $U_k(x) = u_{N-1}x^{N-1} + \cdots + u_1 x + u_0.$

The attack

Say $U_k(x) = u_{N-1}x^{N-1} + \cdots + u_1 x + u_0$.

We assume that we know the binary length $\mathrm{len}_2(u_j) \leq \ell$. Additionally, if $\mathrm{len}_2(u_j) = \ell$ i.e. $u_j = 2^{\ell-1} + y_j 2^{\ell-2} + \cdots$, we assume that we know also y_j.

The attack

Say $U_k(x) = u_{N-1}x^{N-1} + \cdots + u_1 x + u_0$.

We assume that we know the binary length $\mathrm{len}_2(u_j) \le \ell$. Additionally, if $\mathrm{len}_2(u_j) = \ell$ i.e. $u_j = 2^{\ell-1} + y_j 2^{\ell-2} + \cdots$, we assume that we know also y_j.

Then, we can construct an approximation of $U_k(x)$, and for a suitably chosen integer k, we reveal the message **m** by applying a CVP attack to the lattice $\mathcal{L}_k$.

Selection of the approximation vector $\mathbf{E}$

Let the binary expansion $u_j = x_j 2^{\ell-1} + y_j x^{\ell-2} + \cdots$, where $x_j, y_j \in \{0,1\}$ $(0 \le j \le N-1)$, then we set,

$$E_j = \begin{cases} \left.\begin{array}{ll} 2^{\ell-1} + 2^{\ell-2} + 2^{\ell-3}, & \text{if } y_j = 1 \\ 2^{\ell-1} + 2^{\ell-3}, & \text{if } y_j = 0 \end{array}\right| & \text{if } \mathrm{len}_2(u_j) = \ell \text{ (i.e. } x_j = 1) \\ 2^{\ell_j-1} + 2^{\ell_j-2}, \ \text{if } \mathrm{len}_2(u_j) = \ell_j < \ell \text{ (i.e. } x_j = 0) & \end{cases}$$

Selection of the approximation vector $\mathbf{E}$

Let the binary expansion $u_j = x_j 2^{\ell-1} + y_j x^{\ell-2} + \cdots$, where $x_j, y_j \in \{0,1\}$ $(0 \leq j \leq N-1)$, then we set,

$$E_j = \begin{cases} \left.\begin{array}{ll} 2^{\ell-1}+2^{\ell-2}+2^{\ell-3}, & \text{if } y_j = 1 \\ 2^{\ell-1}+2^{\ell-3}, & \text{if } y_j = 0 \end{array}\right| & \text{if } \operatorname{len}_2(u_j) = \ell \text{ (i.e. } x_j = 1) \\ 2^{\ell_j-1}+2^{\ell_j-2}, \text{ if } \operatorname{len}_2(u_j) = \ell_j < \ell \text{ (i.e. } x_j = 0) & \end{cases}$$

We can prove that $|u_j - E_j| \leq 2^{\ell-3} - 1$ and so $\|\mathbf{U}_k - \mathbf{E}\| \leq \sqrt{N}(2^{\ell-3} - 1)$.

Selection of the approximation vector $\mathbf{E}$

Let the binary expansion $u_j = x_j 2^{\ell-1} + y_j x^{\ell-2} + \cdots$, where $x_j, y_j \in \{0,1\}$ $(0 \le j \le N-1)$, then we set,

$$E_j = \begin{cases} \left.\begin{array}{ll} 2^{\ell-1} + 2^{\ell-2} + 2^{\ell-3}, & \text{if } y_j = 1 \\ 2^{\ell-1} + 2^{\ell-3}, & \text{if } y_j = 0 \end{array}\right| & \text{if } \mathrm{len}_2(u_j) = \ell \text{ (i.e. } x_j = 1) \\ 2^{\ell_j-1} + 2^{\ell_j-2}, \ \text{if } \mathrm{len}_2(u_j) = \ell_j < \ell \text{ (i.e. } x_j = 0) \end{cases}$$

We can prove that $|u_j - E_j| \le 2^{\ell-3} - 1$ and so $\|\mathbf{U}_k - \mathbf{E}\| \le \sqrt{N}(2^{\ell-3} - 1)$.

On average we expect $\approx N/2$ coefficients of $U_k(x)$ to have binary length ℓ.

Selection of k

For a moment say we have a fixed value of k

Selection of k

For a moment say we have a fixed value of k

We set $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$

Selection of k

For a moment say we have a fixed value of k

We set $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$

Say $d_1 = \mathrm{dist}(\mathcal{L}_k, \mathbf{t})$ and $d_2 = \mathrm{dist}(\mathbf{U}_k, \mathbf{E})$.

Selection of k

For a moment say we have a fixed value of k

We set $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$

Say $d_1 = \mathrm{dist}(\mathcal{L}_k, \mathbf{t})$ and $d_2 = \mathrm{dist}(\mathbf{U}_k, \mathbf{E})$.

We shall choose k such that $d_1 \approx d_2$.

Selection of k

For a moment say we have a fixed value of k

We set $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$

Say $d_1 = \mathrm{dist}(\mathcal{L}_k, \mathbf{t})$ and $d_2 = \mathrm{dist}(\mathbf{U}_k, \mathbf{E})$.

We shall choose k such that $d_1 \approx d_2$.

We do this by generating NTRU-instances for each k and computing the previous distances. For d_1 we use Babai nearest plain.

Why we pick k such that $d_1 \approx d_2$?

We set $\mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$. This is a lattice point.

Why we pick k such that $d_1 \approx d_2$?

We set $\mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$. This is a lattice point.

Then

$$\mathbf{W} - \mathbf{t} \approx (\mathbf{0}_N, \mathbf{U}_k - \mathbf{E}).$$

Why we pick k such that $d_1 \approx d_2$?

We set $\mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$. This is a lattice point.

Then

$$\mathbf{W} - \mathbf{t} \approx (\mathbf{0}_N, \mathbf{U}_k - \mathbf{E}).$$

That is

$$\|\mathbf{W} - \mathbf{t}\| \approx \|\mathbf{U}_k - \mathbf{E}\| \tag{4}$$

Now, if there is k such that $d_1 = d(\mathcal{L}_k, \mathbf{t}) = ||\mathbf{W} - \mathbf{t}||$ a CVP oracle will (probably) returns $\mathbf{W}$, therefore we can find $\mathbf{m}$.

Why we pick k such that $d_1 \approx d_2$?

We set $\mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$. This is a lattice point.

Then

$$\mathbf{W} - \mathbf{t} \approx (\mathbf{0}_N, \mathbf{U}_k - \mathbf{E}).$$

That is

$$\|\mathbf{W} - \mathbf{t}\| \approx \|\mathbf{U}_k - \mathbf{E}\| \tag{4}$$

Now, if there is k such that $d_1 = d(\mathcal{L}_k, \mathbf{t}) = ||\mathbf{W} - \mathbf{t}||$ a CVP oracle will (probably) returns $\mathbf{W}$, therefore we can find $\mathbf{m}$.

But if $d_1 = d(\mathcal{L}_k, \mathbf{t}) = ||\mathbf{W} - \mathbf{t}||$ then we get, from (4), $d_1 \approx \|\mathbf{U}_k - \mathbf{E}\| = d_2$.

Why we pick k such that $d_1 \approx d_2$?

We set $\mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$. This is a lattice point.

Then

$$\mathbf{W} - \mathbf{t} \approx (\mathbf{0}_N, \mathbf{U}_k - \mathbf{E}).$$

That is

$$\|\mathbf{W} - \mathbf{t}\| \approx \|\mathbf{U}_k - \mathbf{E}\| \tag{4}$$

Now, if there is k such that $d_1 = d(\mathcal{L}_k, \mathbf{t}) = ||\mathbf{W} - \mathbf{t}||$ a CVP oracle will (probably) returns $\mathbf{W}$, therefore we can find $\mathbf{m}$.

But if $d_1 = d(\mathcal{L}_k, \mathbf{t}) = ||\mathbf{W} - \mathbf{t}||$ then we get, from (4), $d_1 \approx \|\mathbf{U}_k - \mathbf{E}\| = d_2$.

For many different k's we computed $d_1 = d(\mathcal{L}_k, \mathbf{t})$ (approximated with Babai) and $d_2 = \|\mathbf{U}_k - \mathbf{E}\|$ where for each instance of NTRU can be computed.

Selection of k

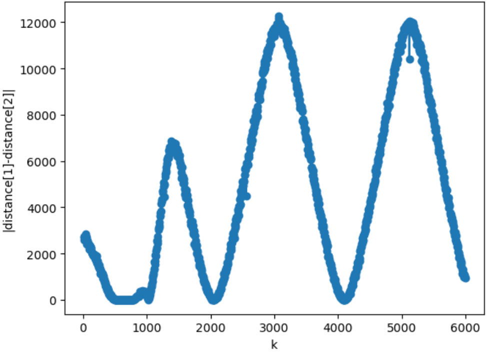

Figure: In this graph we set $q = 2048$. k takes values in the horizontal axis and on the $y-$axis is the $|\mathrm{distance}(\mathbf{U}_k, \mathbf{E}) - \mathrm{distance}(\mathcal{L}_k, \mathbf{t})|$. For each k we generate a new NTRU instance. We remark that Babai's algorithm provides outputs with distances close to $\mathrm{distance}(\mathbf{U}_k, \mathbf{E})$ for $k \in [520, 790]$. We finally select k to be 550.

Now having a way to select both $\mathbf{E}$ and k we can execute our attack.

Now having a way to select both $\mathbf{E}$ and k we can execute our attack. We applied it for the three variants of NTRU-HPS, namely ntruhps2048509, ntruhps2048677 and ntruhps4096821. For all the experiments we revealed the unknown message. The attack time was negligible, approximately 1 second.

Thank you!