Message Recovery in NTRU Encryption 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, alongside the digital signature scheme Falcon.

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, alongside the digital signature scheme Falcon.

We believe that NTRU based schemes are post quantum secure.

- It is 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

- It is 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

- It is 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

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[4], where if we know some bits of many ephemeral keys we can compute the secret key,

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

[5] 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[4], 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 modulus RSA,

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

[5] 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[4], 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 modulus RSA,
- and more recently an attack to kyber[5] where if we know some information about the LWE secret is leaked through hints, modeled as inner products with known vectors, we compute the secret key.

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

[5] 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 N is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^\ell$).

Introduction to NTRU

In all the NTRU variants N is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^{\ell}$).

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 N is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^{\ell}$).

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 N is a prime number and in NTRU-HPS/HRSS, q is a power of 2 (say $q = 2^{\ell}$).

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 $(f(x), g(x)) \stackrel{\$}{\leftarrow} \mathcal{M}_f \times \mathcal{M}_g$.

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 $(f(x), g(x))$ is referred to as the *search NTRU problem*.

Introduction to NTRU

The secret key (f, g) 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 key (f, g) 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 part $\mathbf{C}(\mathbf{h})$, is the cyclic matrix generated by $\mathbf{h} = (h_0, ..., h_{N-1})$, for $h(x) = h_{N-1}x^{N-1} + \cdots + h_1 x + h_0$.

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

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

Lattice $\mathcal{L}_k$, also contains the vector $(\mathbf{r}, \mathbf{c} - \mathbf{m})$, where $\mathbf{r}$ is the nonce, $\mathbf{c}$ the encryption of $\mathbf{m}$. So, using a suitable target vector we can implement a message recovery attack.

Our attack

In more details, we multiply the encryption equation by an integer k (we shall calculate later), so we get

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

Our attack

In more details, we multiply the encryption equation by an integer k (we shall calculate 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 $U_k(x)$ is equivalent to knowing $m(x)$.

Our attack

In more details, we multiply the encryption equation by an integer k (we shall calculate 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 $U_k(x)$ is equivalent to knowing $m(x)$.

Our attack is a *message recovery attack*, and reveals $m(x)$ assuming an approximation of $U_k(x)$.

Our attack

Say $U_k(x) = u_{N-1}x^{N-1} + \cdots + c_1 x + c_0.$

Our attack

Say $U_k(x) = u_{N-1}x^{N-1} + \cdots + c_1 x + c_0$.
Also $q = 2^{\ell}$ and since all the u_j are mod q i.e. $u_j < q$ their binary lengths will be $\leq \ell$.

Our attack

Say $U_k(x) = u_{N-1}x^{N-1} + \cdots + c_1x + c_0$.

Also $q = 2^\ell$ and since all the u_j are $\mod q$ i.e. $u_j < q$ their binary lengths will be $\leq \ell$.

When $\mathrm{len}_2(u_j) = \ell$ i.e. u_j has length ℓ, we assume that we know the coefficients of $2^{\ell-1}$ and $2^{\ell-2}$ in the binary expansion of u_j.

Our attack

Say $U_k(x) = u_{N-1}x^{N-1} + \cdots + c_1x + c_0$.

Also $q = 2^\ell$ and since all the u_j are $\mod q$ i.e. $u_j < q$ their binary lengths will be $\leq \ell$.

When $\mathrm{len}_2(u_j) = \ell$ i.e. u_j has length ℓ, we assume that we know the coefficients of $2^{\ell-1}$ and $2^{\ell-2}$ in the binary expansion of u_j.

Otherwise, we assume we know $\mathrm{len}_2(u_j) = \ell_j < \ell$.

Then our attack recovers the message $m(x)$

Main idea of the attack

We work in 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 in 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 in 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.

We shall see that a nice approximation of $\mathbf{U}_k$ reveals $\mathbf{m}$.

Main idea of the attack

- Say that we know an approximation $\mathbf{E}$ of $\mathbf{u}_k$. Set the target vector $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$, and $\mathbf{w} \leftarrow \mathrm{CVP}(\mathcal{L}_k, \mathbf{t})$

Main idea of the attack

- Say that we know an approximation $\mathbf{E}$ of $\mathbf{u}_k$. Set the target vector $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$, and $\mathbf{w} \leftarrow \mathrm{CVP}(\mathcal{L}_k, \mathbf{t})$
- Also $\mathbf{0}_N \approx -\mathbf{m}$ and $(\mathbf{b}_k + \mathbf{U}_k) \approx (\mathbf{b}_k + \mathbf{E})$ so $\mathbf{t} \approx \mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$

Main idea of the attack

- Say that we know an approximation $\mathbf{E}$ of $\mathbf{u}_k$. Set the target vector $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$, and $\mathbf{w} \leftarrow \mathrm{CVP}(\mathcal{L}_k, \mathbf{t})$
- Also $\mathbf{0}_N \approx -\mathbf{m}$ and $(\mathbf{b}_k + \mathbf{U}_k) \approx (\mathbf{b}_k + \mathbf{E})$ so $\mathbf{t} \approx \mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$
- We remark that $\|\mathbf{w} - \mathbf{W}\| < 2\|\mathbf{W} - \mathbf{t}\|$.

Main idea of the attack

- Say that we know an approximation $\mathbf{E}$ of $\mathbf{u}_k$. Set the target vector $\mathbf{t} = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E})$, and $\mathbf{w} \leftarrow \mathrm{CVP}(\mathcal{L}_k, \mathbf{t})$
- Also $\mathbf{0}_N \approx -\mathbf{m}$ and $(\mathbf{b}_k + \mathbf{U}_k) \approx (\mathbf{b}_k + \mathbf{E})$ so $\mathbf{t} \approx \mathbf{W} = (-\mathbf{m}, \mathbf{U}_k + \mathbf{b}_k)$
- We remark that $\|\mathbf{w} - \mathbf{W}\| < 2\|\mathbf{W} - \mathbf{t}\|$.
- If $\|\mathbf{W} - \mathbf{t}\| < \lambda_1/2$ and the fact that $\mathbf{W}, \mathbf{w} \in \mathcal{L}_k$ we get $\mathbf{w} = \mathbf{W}$. Since $\mathbf{W}_{[1,N]} = -\mathbf{m}$, we retrieve $\mathbf{m}$.

Main idea of the attack

For λ_1, we have the following result.

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, $\lambda_1(\mathcal{L}_k) = \sqrt{k^2+1}$.

Selection of **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,

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

Selection of **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,

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

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

Selection of 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 $\|\mathbf{u}_k - \mathbf{E}\| \leq \sqrt{N}(2^{\ell-3} - 1)$.

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

Selection of k

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

Selection of k

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

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 $d_1 \approx d_2$?

Set $\mathbf{t}' = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{u}_k)$, we have chosen $\mathbf{E}$ such that $\mathbf{E} \approx \mathbf{u}_k$, so $\mathbf{t}' \approx (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E}) = \mathbf{t}$.

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

Set $\mathbf{t}' = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{u}_k)$, we have chosen $\mathbf{E}$ such that $\mathbf{E} \approx \mathbf{u}_k$, so $\mathbf{t}' \approx (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E}) = \mathbf{t}$.

There is a lattice point $\mathbf{W} = (-\mathbf{m}, \mathbf{u}_k + \mathbf{b}_k)$ such that

$$\mathbf{W} - \mathbf{t} \approx \mathbf{t}' - \mathbf{t} = (\mathbf{0}_N, \mathbf{u}_k - \mathbf{E}).$$

That is $\|\mathbf{W} - \mathbf{t}\| \approx \|\mathbf{u}_k - \mathbf{E}\|$.

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

Set $\mathbf{t}' = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{u}_k)$, we have chosen $\mathbf{E}$ such that $\mathbf{E} \approx \mathbf{u}_k$, so $\mathbf{t}' \approx (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E}) = \mathbf{t}$.

There is a lattice point $\mathbf{W} = (-\mathbf{m}, \mathbf{u}_k + \mathbf{b}_k)$ such that

$$\mathbf{W} - \mathbf{t} \approx \mathbf{t}' - \mathbf{t} = (\mathbf{0}_N, \mathbf{u}_k - \mathbf{E}).$$

That is $\|\mathbf{W} - \mathbf{t}\| \approx \|\mathbf{u}_k - \mathbf{E}\|$.

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

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

Set $\mathbf{t}' = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{u}_k)$, we have chosen $\mathbf{E}$ such that $\mathbf{E} \approx \mathbf{u}_k$, so $\mathbf{t}' \approx (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E}) = \mathbf{t}$.

There is a lattice point $\mathbf{W} = (-\mathbf{m}, \mathbf{u}_k + \mathbf{b}_k)$ such that

$$\mathbf{W} - \mathbf{t} \approx \mathbf{t}' - \mathbf{t} = (\mathbf{0}_N, \mathbf{u}_k - \mathbf{E}).$$

That is $\|\mathbf{W} - \mathbf{t}\| \approx \|\mathbf{u}_k - \mathbf{E}\|$.

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

But $d(\mathcal{L}_k, \mathbf{t}) = ||\mathbf{W} - \mathbf{t}||$ provides $d(\mathcal{L}_k, \mathbf{t}) \approx \|\mathbf{u}_k - \mathbf{E}\| = d_2$.

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

Set $\mathbf{t}' = (\mathbf{0}_N, \mathbf{b}_k + \mathbf{u}_k)$, we have chosen $\mathbf{E}$ such that $\mathbf{E} \approx \mathbf{u}_k$, so $\mathbf{t}' \approx (\mathbf{0}_N, \mathbf{b}_k + \mathbf{E}) = \mathbf{t}$.

There is a lattice point $\mathbf{W} = (-\mathbf{m}, \mathbf{u}_k + \mathbf{b}_k)$ such that

$$\mathbf{W} - \mathbf{t} \approx \mathbf{t}' - \mathbf{t} = (\mathbf{0}_N, \mathbf{u}_k - \mathbf{E}).$$

That is $\|\mathbf{W} - \mathbf{t}\| \approx \|\mathbf{u}_k - \mathbf{E}\|$.

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

But $d(\mathcal{L}_k, \mathbf{t}) = ||\mathbf{W} - \mathbf{t}||$ provides $d(\mathcal{L}_k, \mathbf{t}) \approx \|\mathbf{u}_k - \mathbf{E}\| = d_2$.

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

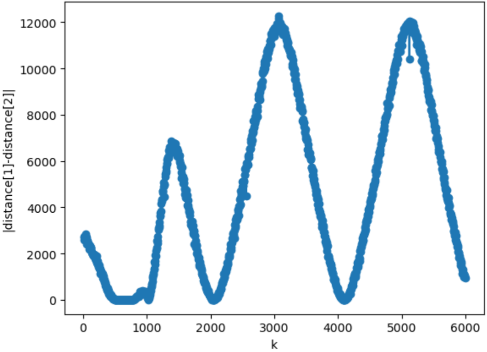

Figure: In this graph we set $q = 2048$. k takes values in the horizontal axis and on the $y-$axis is the $|\mathrm{distance}(\mathbf{u}_k, \mathbf{E}) - \mathrm{distance}(\mathcal{L}_k, \mathbf{t})|$. For each k we generate a new NTRU instance. We remark that Babai's algorithm provides outputs with 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. From the previous analysis we pick $k = 550$.

Now having a way to select both **E** and k we can execute our attack. From the previous analysis we pick $k = 550$.

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!