# NTRU and BLISS

Tanja Lange

Technische Universiteit Eindhoven

21 March 2017

# Commercial Break

**Summer school on post-quantum crypto**
Eindhoven, 19–23 June 2017
https://2017.pqcrypto.org/school/index.html

**Executive school on post-quantum crypto**
Eindhoven, 22–23 June 2017
https://2017.pqcrypto.org/exec/index.html

**PQCrypto 2017**
Utrecht, 26–28 June 2017
https://2017.pqcrypto.org/conference/index.html

# NTRU and BLISS

Tanja Lange

Technische Universiteit Eindhoven

21 March 2017

# NTRU

- Introduced by Hoffstein–Pipher–Silverman in 1998.
- Security related to lattice problems; pre-version cryptanalyzed with LLL by Coppersmith and Shamir.
- System parameters $(n, q)$, $n$ prime, integer $q$, $\gcd(3, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^n - 1)$.

# NTRU

- Introduced by Hoffstein–Pipher–Silverman in 1998.
- Security related to lattice problems; pre-version cryptanalyzed with LLL by Coppersmith and Shamir.
- System parameters $(n, q)$, $n$ prime, integer $q$, $\gcd(3, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^n - 1)$.
- Private key: $f, g \in R$ sparse with coefficients in $\{-1, 0, 1\}$. Additional requirement: $f$ must be invertible in $R$ modulo $q$.
- Public key $h = 3g/f \bmod q$.
- Can see this as lattice with basis matrix

$$B = \begin{pmatrix} q\, I_n & 0 \\ H & I_n \end{pmatrix},$$

  where $H$ corresponds to multiplication by $h/3$ modulo $x^n - 1$.
- $(g, f)$ is a short vector in the lattice as result of

$$(k, f)B = (kq + f \cdot h/3, f) = (g, f)$$

  for some polynomial $k$ (from $fh/3 = g - kq$).

# Classic NTRU

- System parameters $(n, q)$, $n$ prime, integer $q$, $\gcd(3, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^n - 1)$, some use additional reduction modulo $q$, ring denoted by $R_q$.

# Classic NTRU

- System parameters $(n, q)$, $n$ prime, integer $q$, $\gcd(3, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^n - 1)$, some use additional reduction modulo $q$, ring denoted by $R_q$.
- Private key: $f, g \in R$ with coefficients in $\{-1, 0, 1\}$, almost all coefficients are zero (small fixed number are nonzero).
  Additional requirement: $f$ must be invertible in $R$ modulo $q$ and modulo 3.
- Public key $h = 3g/f \bmod q$.

# Classic NTRU

- System parameters $(n, q)$, $n$ prime, integer $q$, $\gcd(3, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^n - 1)$, some use additional reduction modulo $q$, ring denoted by $R_q$.
- Private key: $f, g \in R$ with coefficients in $\{-1, 0, 1\}$, almost all coefficients are zero (small fixed number are nonzero).
  Additional requirement: $f$ must be invertible in $R$ modulo $q$ and modulo 3.
- Public key $h = 3g/f \bmod q$.
- Encryption of message $m \in R$, coefficients in $\{-1, 0, 1\}$:
  Pick random, sparse $r \in R$, same sample space as $f$; compute:

$$c = r \cdot h + m \bmod q.$$

- Decryption of $c \in R_q$: Compute

$$a = f \cdot c = f(rh + m) \equiv f(3rg/f + m) \equiv 3rg + fm \bmod q,$$

  move all coefficients to $[-q/2, q/2]$. If everything is small enough then $a$ equals $3rg + fm$ in $R$ and $m = a/f \bmod 3$.

# Decryption failures

Decryption of $c \in R_q$: Compute

$$a = f \cdot c = f(rh + m) \equiv f(3rg/f + m) \equiv 3rg + fm \bmod q,$$

move all coefficients to $[-q/2, q/2]$. If everything is small enough then $a$ equals $3rg + fm$ in $R$ and $m = a/f \bmod 3$.
Let

$$L(d, t) = \{F \in R | F \text{ has } d \text{ coefficients equal to } 1 \text{ and } t \text{ coefficients equal to } -1, \text{all others } 0\}.$$

Let $f \in L(d_f, d_f - 1)$, $r \in L(d_r, d_r)$, and $g \in L(d_g, d_g)$ with $d_r < d_g$.
Then $3rg + fm$ has coefficients of size at most

$$3 \cdot 2d_r + 2d_f - 1$$

which is larger than $q/2$ for typical parameters. Such large coefficients are highly unlikely – but annoying for applications and guarantees.
Security decreases with large $q$; reduction is important.

# Evaluation-at-1 attack

Ciphertext equals $c = rh + m$ and $r \in L(d_r, d_r)$, so $r(1) = 0$ and $g \in L(d_g, d_g)$, so $h(1) = g(1)/f(1) = 0$.

This implies

$$c(1) = r(1)h(1) + m(1) = m(1)$$

which gives information about $m$, in particular if $|m(1)|$ is large.

NTRU rejects extreme messages – this is dealt with by randomizing $m$ via a padding (not mentioned so far).

For other choices of $r$ and $h$, such as $L(d_r, d_r - 1)$ or such, one knows $r(1)$ and $h$ is public, so evaluation at 1 leaks $m(1)$.

# Evaluation-at-1 attack

Ciphertext equals $c = rh + m$ and $r \in L(d_r, d_r)$, so $r(1) = 0$ and $g \in L(d_g, d_g)$, so $h(1) = g(1)/f(1) = 0$.

This implies

$$c(1) = r(1)h(1) + m(1) = m(1)$$

which gives information about $m$, in particular if $|m(1)|$ is large.

NTRU rejects extreme messages – this is dealt with by randomizing $m$ via a padding (not mentioned so far).

For other choices of $r$ and $h$, such as $L(d_r, d_r - 1)$ or such, one knows $r(1)$ and $h$ is public, so evaluation at 1 leaks $m(1)$.

Could also replace $x^n - 1$ by $\Phi_n = (x^n - 1)/(x - 1)$ to avoid attack.

# Mathematical attacks

- Meet-in-the-middle attack;
- Lattice-basis reduction (e.g. LLL, BKZ);
- Hybrid attack, combining both.

Crypto attacks:

- Chosen-ciphertext attacks;
- Decryption-failure attacks;
- Complicated padding systems.

# Odlyzko's meet-in-the-middle attack on NTRU

- Idea: split the possibilities for $f$ in two parts

$$h = (f_1 + f_2)^{-1} 3g$$
$$f_1 \cdot h = 3g - f_2 \cdot h.$$

- If there was no $g$: collision search in $f_1 \cdot h$ and $-f_2 \cdot h$

# Odlyzko's meet-in-the-middle attack on NTRU

- Idea: split the possibilities for $f$ in two parts

$$h = (f_1 + f_2)^{-1} 3g$$
$$f_1 \cdot h = 3g - f_2 \cdot h.$$

- If there was no $g$: collision search in $f_1 \cdot h$ and $-f_2 \cdot h$
- Solution: look for collisions in $c(f_1 \cdot h)$ and $c(-f_2 \cdot h)$ with

$$c(a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}) = (\mathbf{1}(a_0 > 0), \ldots, \mathbf{1}(a_{n-1} > 0))$$

  using that $g$ is small and thus $+g$ often does not change the sign.
- If $c(f_1 \cdot h) = c(-f_2 \cdot h)$ check whether $h(f_1 + f_2)$ is in $L(d_g, d_g)$.
- Basically runs in squareroot of size of search space.

# Odlyzko's meet-in-the-middle attack on NTRU

- Idea: split the possibilities for $f$ in two parts

$$h = (f_1 + f_2)^{-1} 3g$$
$$f_1 \cdot h = 3g - f_2 \cdot h.$$

- If there was no $g$: collision search in $f_1 \cdot h$ and $-f_2 \cdot h$
- Solution: look for collisions in $c(f_1 \cdot h)$ and $c(-f_2 \cdot h)$ with

$$c(a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}) = (\mathbf{1}(a_0 > 0), \ldots, \mathbf{1}(a_{n-1} > 0))$$

  using that $g$ is small and thus $+g$ often does not change the sign.
- If $c(f_1 \cdot h) = c(-f_2 \cdot h)$ check whether $h(f_1 + f_2)$ is in $L(d_g, d_g)$.
- Basically runs in squareroot of size of search space.
- General running time / memory mitm (Christine van Vredendaal)

$$L = \sqrt{|S|}/\sqrt{s}.$$

# Attackable rotations

In NTRU, $x^i f$ is simply a rotation of $f$, so it has the same coefficients, just at different positions. This means, $x^i f$ also gives a solution in the mitm attack: $hx^i f = x^i g$ has same sparsity etc., increasing the number of targets.
Decryption using $x^i f$ works the same as with $f$ for NTRU, so each target is valid.

# Security against Odlyzko's meet-in-the-middle attack

- Number of choices for $f$ is

$$\binom{n}{t}\binom{n-t}{t-1}$$

  because $f$ has $2t-1$ non-zero coefficients.
- Number of rotations is $n$.
- Running time / memory against NTRU

$$L = \frac{\sqrt{\binom{n}{t}\binom{n-t}{t-1}}}{\sqrt{n}}.$$

# Security against Odlyzko's meet-in-the-middle attack

- Number of choices for $f$ is

$$\binom{n}{t}\binom{n-t}{t-1}$$

because $f$ has $2t-1$ non-zero coefficients.

- Number of rotations is $n$.
- Running time / memory against NTRU

$$L = \frac{\sqrt{\binom{n}{t}\binom{n-t}{t-1}}}{\sqrt{n}}.$$

- Memory requirement can be reduced.

# Security against lattice sieving

- Recall $h = 3g/f$ in $\mathcal{R}/q$.
- This implies that for $k \in \mathcal{R}$: $f \cdot h/3 + k \cdot q = g$.
- NTRU lattice

$$\begin{pmatrix} k & f \end{pmatrix} \begin{pmatrix} qI_n & 0 \\ H & I_n \end{pmatrix} = \begin{pmatrix} g & f \end{pmatrix}.$$

# Security against lattice sieving

- Recall $h = 3g/f$ in $\mathcal{R}/q$.
- This implies that for $k \in \mathcal{R}$: $f \cdot h/3 + k \cdot q = g$.
- NTRU lattice

$$\begin{pmatrix} k & f \end{pmatrix} \begin{pmatrix} qI_n & 0 \\ H & I_n \end{pmatrix} = \begin{pmatrix} g & f \end{pmatrix}.$$

- Keypair $(g, f)$ is a short vector in this lattice.
- Asymptotically sieving works in $2^{0.292 \cdot 2p + o(p)}$ using $2^{0.208 \cdot 2p + o(p)}$ memory.
- Crossover point between sieving and BKZ is still unclear.
- Memory is more an issue than time.

# Hybrid attack

Howgrave-Graham combines lattice basis reduction and meet-in-the-middle attack.

- Idea: reduce submatrix of the NTRU lattice, then perform mitm on the rest.

# Hybrid attack

Howgrave-Graham combines lattice basis reduction and meet-in-the-middle attack.

- Idea: reduce submatrix of the NTRU lattice, then perform mitm on the rest.
- Use BKZ on submatrix $B$ to get $B'$:

$$C \cdot \begin{pmatrix} qI_n & 0 \\ H & I_n \end{pmatrix} = \left(\begin{array}{ccc} qI_w & 0 & 0 \\ \hline * & B' & 0 \\ \hline * & * & I_{w'} \end{array}\right).$$

- Guess options for last $w'$ coordinates of $f$, using collision search (as before).
- If the Hermite factor of $B'$ is small enough, then a rounding algorithm can detect collision of halfguesses.

# Security against the hybrid attack

- Balance the costs of the BKZ and mitm phase.

# Security against the hybrid attack

- Balance the costs of the BKZ and mitm phase.
- Hoffstein, Pipher, Schanck, Silverman, Whyte, and Zhang [HPSWZ15] published simplfied analyzis tool.
- Compute BKZ costs with Chen-Nguyen simulator.
- Estimate the mitm costs by estimating the size of the projected space [HPSWZ15].

# How about other interesting candidates?

- Bimodal Lattice Signature Scheme (BLISS) (CRYPTO '13 by Léo Ducas and Alain Durmus and Tancrède Lepoint and Vadim Lyubashevsky)
- Pretty short and efficient; already included in strongSwan (library for IPsec-based VPN).
- Needs noise from discrete Gaussian distribution.
- Security is related to lattice-based problem; direct reduction to $\mathrm{SIS}_q$ = Short Integer Solution mod q.

# Background

- Work in $R = \mathbf{Z}[x]/(x^n + 1)$, $n = 2^r$, and $R_q = (\mathbf{Z}/q)[x]/(x^n + 1)$ for $q$ prime.
- Switch representation between polynomial and vector notation.

$$f(x) = \sum_{i=0}^{n-1} f_i x^i \Leftrightarrow f = (f_{n-1}, f_{n-2}, \ldots, f_1, f_0).$$

- Polynomial multiplication then corresponds to vector-matrix multiplication. Let $f, g, \in R_q$, then

$$f \cdot g = fG = gF,$$

where $F, G \in (\mathbf{Z}/q)^{n \times n}$ match vectors of $x^i f$ and $x^j g$.

$$\begin{pmatrix} f_0 & -f_{n-1} & -f_{n-2} & \ldots & -f_1 \\ f_1 & f_0 & -f_{n-1} & \ldots & -f_2 \\ \vdots & \vdots & \ddots & \vdots & \\ f_{n-1} & f_{n-2} & f_{n-3} & \ldots & f_0 \end{pmatrix}$$

# Simplified BLISS

- Secret key $S = (s_1, s_2) = (f, 2g + 1) \in R_q^2$, $f, g$ sparse in $\{0, \pm 1\}^n$.
- Public key $A = (a_1, a_2) \in R_{2q}^2$, with key equation $a_1 s_1 + a_2 s_2 \equiv q \bmod 2q$.
- Computed as $a_q = (2g + 1)/f \bmod q$ (restart if $f$ is not invertible); then $A = (2a_q, q - 2) \bmod 2q$.

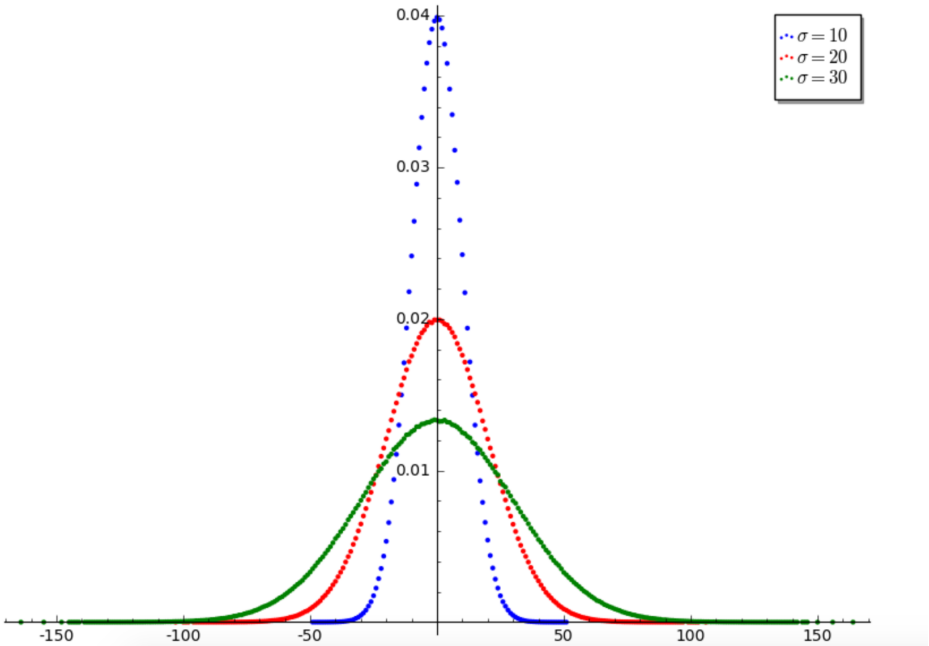
# Simplified BLISS

- Secret key $S = (s_1, s_2) = (f, 2g + 1) \in R_q^2$, $f, g$ sparse in $\{0, \pm 1\}^n$.
- Public key $A = (a_1, a_2) \in R_{2q}^2$, with key equation $a_1 s_1 + a_2 s_2 \equiv q \bmod 2q$.
- Computed as $a_q = (2g + 1)/f \bmod q$ (restart if $f$ is not invertible); then $A = (2a_q, q - 2) \bmod 2q$.
- $2a_q s_1 + (q - 2)s_2 \equiv 2(2g + 1)/f \cdot f + (q - 2)(2g + 1) \equiv q \bmod 2q$.
- Attacker can verify key guess for $f$ with key equation; $g$ computable; $-S$ just as good as $S$.

# Simplified BLISS

- Secret key $S = (s_1, s_2) = (f, 2g+1) \in R_q^2$, $f, g$ sparse in $\{0, \pm 1\}^n$.
- Public key $A = (a_1, a_2) \in R_{2q}^2$, with key equation $a_1 s_1 + a_2 s_2 \equiv q \bmod 2q$.
- Computed as $a_q = (2g+1)/f \bmod q$ (restart if $f$ is not invertible); then $A = (2a_q, q-2) \bmod 2q$.
- $2a_q s_1 + (q-2)s_2 \equiv 2(2g+1)/f \cdot f + (q-2)(2g+1) \equiv q \bmod 2q$.
- Attacker can verify key guess for $f$ with key equation; $g$ computable; $-S$ just as good as $S$.
- To sign $m$, sample $y$ from discrete $n$-dim Gaussian $D_\sigma^n$.
- $c = H(Ay \bmod 2q, m)$ // $H$ special sparse hash function.
- Signature: $(z, c)$ with $z = y + (-1)^b s_1 \cdot c \bmod 2q$. // $b$ random
  Algorithm uses rejection sampling so that it does not leak $s_1$.
- Accept signature if $z$ is short and $c = H(Az + qc \bmod 2q, m)$.
  Works: $Az + qc \equiv A(y + (-1)^b Sc) + qc \equiv A(y + (-1)^b s_1 c) + qc \equiv Ay + ((-1)^b AS + q) \equiv Ay \bmod 2q$

# Discrete Gaussian




- Step 1 in signature algorithm: $y \leftarrow D_\sigma^m$
- This is required to achieve (provable) security and small signature size.
- Not straightforward to do in practie: high precision required.
- Side-channel attack on sampling gives (part of) $y$.
- Can get $\pm s_1 = (z - y)/c \in R_q$ if we know $y$, the error vector/polynomial; ($c$ needs to be invertible).
- Full details in `https://eprint.iacr.org/2016/300` (with Groot Bruinderink, Hülsing, and Yarom).