Round5: a KEX based on learning with rounding over the rings

Zhenfei Zhang

zhenfei.zhang@hotmail.com

September 19, 2018

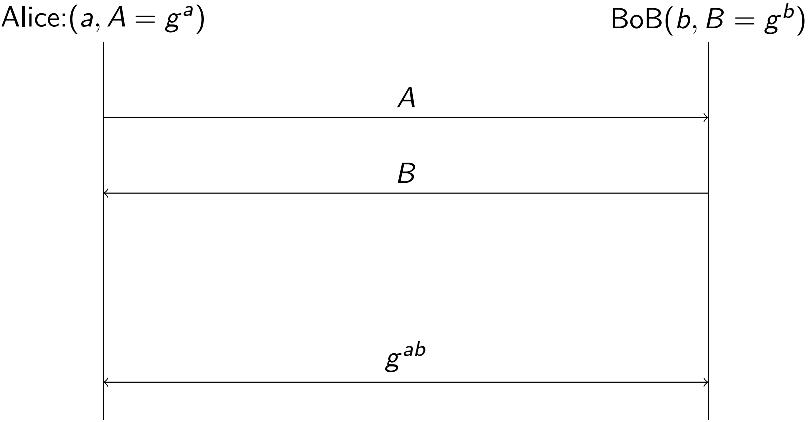
Areas I have been working on

- Theoretical results
  - Signature schemes: Falcon (NTRUSign+GPV), pqNTRUSign
  - Security proofs: Computational R-LWR problem
  - Fully homomorphic encryptions
  - Raptor: lattice based linkable ring signature (Blockchains!)
- Practical instantiations
  - NTRU, Round5
  - Cryptanalysis and parameter derivation for lattices
  - Efficient implementations: AVX-2
  - Constant time implementations
- Standardization: NIST, IETF, ETSI, ISO, CACR PQC process
- Deployment: enabling PQC for TLS, Tor, libgcrypt
- Under the radar: lattice based DAA, NIZK

This talk

- Theoretical results
  - Signature schemes: Falcon (NTRUSign+GPV), pqNTRUSign
  - Security proofs: Computational R-LWR problem
  - Fully homomorphic encryptions
  - Raptor: lattice based linkable ring signature (Blockchains!)
- Practical instantiations
  - NTRU, Round5
  - Cryptanalysis and parameter derivation for lattices
  - Efficient implementations: AVX-2
  - Constant time implementations
- Standardization: NIST, IETF, ETSI, ISO, CACR PQC process
- Deployment: enabling PQC for TLS, Tor, libgcrypt
- Under the radar: lattice based DAA, NIZK

Diffie-Hellman



- A, B and g^{ab} are group elements over $\mathbb{Z}_q^*$.

Alice:$(a, A = Ga + e_a)$ BoB$(b, B = Gb + e_b)$

$$\xrightarrow{A}$$

$$\xleftarrow{B}$$

$$\xleftrightarrow{aGb + ae_b \approx bGa + be_a}$$

- Every element is a ring element over $\mathcal{R} := \mathbb{Z}_q[x]/f(x)$.

RLWE-KEX

Alice:$(a, X = Ga + e_a)$ BoB$(b, B = Gb + e_b)$

$$\xrightarrow{\quad A \quad}$$

$$\xleftarrow{\quad B \quad}$$

$$\xleftarrow{\quad \text{Reconciliation } r \quad}$$

$$\textsc{Extract}(aGb + ae_b, r) = \textsc{Extract}(bGa + be_a, r)$$

- Every element is a ring element over $\mathcal{R} := \mathbb{Z}_q[x]/f(x)$.

NTRU-KEM

Alice:$(a = 2t + 1, A = 2g/a)$ BoB:(b, m)

$$\xrightarrow{\quad A \quad}$$

$B = 2bA + m$

$$\xleftarrow{\quad B \quad}$$

$$\begin{aligned} m = Ba &= 2bg + ma \\ &= m \bmod 2 \end{aligned}$$

$$\xleftrightarrow{\quad \text{m} \quad}$$

- Every element is a ring element over $\mathcal{R} := \mathbb{Z}_q[x]/(x^N - 1)$.

NTRU-KEM vs RLWE-KEX

	NTRU	R-LWE
Ring	$\mathbb{Z}_q[x]/(x^N - 1)$	$\mathbb{Z}_q[x]/(x^N + 1)$
Provable security	No	Yes
Secrets	Trinary: $\{-1, 0, 1\}^{\text{dim}}$	Gaussian: $\chi_{\sqrt{q}}^{\text{dim}}$
Errors	Rounded, binary	Gaussian: $\chi_{\sqrt{q}}^{\text{dim}}$
Trapdoor	Yes	No
KeyGen	Slow	Fast
CT size	Small	Large

Major concerns on NTRU

- No provable security
- Slow key generation c.f. New Hope

NTRU-KEM vs RLWE-KEX

Major concerns on NTRU

- No provable security
- Slow key generation c.f. New Hope

NTRU-KEM vs RLWE-KEX

	NTRU	R-LWE
Ring	$\mathbb{Z}_q[x]/(x^N - 1)$	$\mathbb{Z}_q[x]/(x^N + 1)$
Provable security	No	Yes
Secrets	Trinary: $\{-1, 0, 1\}^{\mathrm{dim}}$	Gaussian: $\chi_{\sqrt{q}}^{\mathrm{dim}}$
Errors	Rounded, binary	Gaussian: $\chi_{\sqrt{q}}^{\mathrm{dim}}$
Trapdoor	Yes	No
KeyGen	Slow	Fast
CT size	Small	Large

- RLWE is hard for any ring of integers [PRS17]

NTRU-KEM vs RLWE-KEX

	NTRU	R-LWE
Ring	$\mathbb{Z}_q[x]/(\phi_N(x))$	$\mathbb{Z}_q[x]/(x^N+1)$
Provable security	Yes(?)	Yes
Secrets	Trinary: $\{-1,0,1\}^{\text{dim}}$	Gaussian: $\chi_{\sqrt{q}}^{\text{dim}}$
Errors	Rounded, binary	Gaussian: $\chi_{\sqrt{q}}^{\text{dim}}$
Trapdoor	Yes	No
KeyGen	Slow	Fast
CT size	Small	Large

- RLWE is hard for any ring of integers [PRS17]

NTRU-KEM vs RLWE-KEX

	NTRU	R-LWE
Ring	$\mathbb{Z}_q[x]/(\phi_N(x))$	$\mathbb{Z}_q[x]/(x^N+1)$
Provable security	Yes(?)	Yes
Secrets	Trinary: $\{-1,0,1\}^{\text{dim}}$	Gaussian: $\chi_{\sqrt{q}}^{\text{dim}}$
Errors	Rounded, binary	Gaussian: $\chi_{\sqrt{q}}^{\text{dim}}$
Trapdoor	No	No
KeyGen	Slow	Fast
CT size	Small	Large

- RLWE is hard for any ring of integers [PRS17]
- Hardness of dec R-LWR is an open problem

NTRU-KEM vs RLWE-KEX

	NTRU	R-LWE
Ring	$\mathbb{Z}_q[x]/(\phi_N(x))$	$\mathbb{Z}_q[x]/(x^N+1)$
Provable security	Yes	Yes
Secrets	Trinary: $\{-1,0,1\}^{\text{dim}}$	Gaussian: $\chi^{\text{dim}}_{\sqrt{q}}$
Errors	Rounded, binary	Gaussian: $\chi^{\text{dim}}_{\sqrt{q}}$
Trapdoor	No	No
KeyGen	Fast	Fast
CT size	Small	Large

- RLWE is hard for any ring of integers [PRS17]
- Hardness of dec R-LWR is an open problem (more on this later)

The new *NTRU*, a.k.a. R-LWR-KEX

Alice:$(a, A = Round(Ga))$ BoB$(b, B = Round(Gb))$

$\xrightarrow{\quad A \quad}$

$\xleftarrow{\quad B \quad}$

$\xleftarrow{\text{Reconciliation } r}$

$\textsc{Extract}(Round(aB), r) = \textsc{Extract}(Round(bA), r)$

- a, b are ring elements over $\mathcal{R} := \mathbb{Z}_q[x]/f(x)$;
- A, B are rounded over $\mathbb{Z}_p[x]$.

R-LWR-KEX

Improvements

- Prime cyclotomic ring, i.e., $\phi_{743}(x) = (x^{743} - 1)/(x - 1)$
- Rounding instead of errors

"*Disadvantages*"

- Parameters not compatible with number theoretic transform (NTT)
- Noise dependency

Improvements I - PC ring

Prime Cyclotomic ring

- PC ring as secure as power-of-2 cyclotomics;
  - i.e., $\phi_{2048}(x) = x^{1024} + 1$;
- Degree $\approx$ 700 offers enough security against BKZ attacks with quantum sieving;
- NewHope - has to be 512, 1024, etc.;
- Kyber - a multiple of 256;
- PC - any prime > 700;
  - Also used in LIMA, NTRU-KEM, etc.


ONE
RING
TO
RULE
THEM
ALL

Improvements II - Rounding

Rounding

- Less randomness sampling - e_a and e_b;
- Ciphertext reduced to $n \log p$, c.f. $n \log q$;
- Small enough to be in an MTU for TLS;
- Introduces new assumptions.

"*Disadvantages*" I - Ring multiplications

Rule of Thumb

School book $\gg$ Karatsuba/Toom-Cook $\gtrapprox$ NTT $>$ Index based

"*Disadvantages*" I - Ring multiplications

Rule of Thumb

School book $\gg$ Karatsuba/Toom-Cook $\gtrapprox$ NTT $>$ Index based

Karatsuba and Toom-Cook

- Divide and Conquer;
- Parameter dependent optimizations;
  - Improving NTRU-743 reference implementation by 2.3x;
- Constant time; strong side channel resistance;
- Slightly slower than NTT for similar N.

"*Disadvantages*" I - Ring multiplications

Rule of Thumb

School book $\gg$ Karatsuba/Toom-Cook $\gtrapprox$ NTT $>$ Index based

Index based

- Super friendly with a trinary polynomial;
- Even faster than NTT;
- Constant time iff $\textsc{Ham}(a)$ and $\textsc{Ham}(b)$ are constant;
- Memory leakage.

"*Disadvantages*" II - Noise management

Rational: Use ECC to control errors

- Consider $c(x) = a(x)b(x) \bmod x^{N-1} + x^{N-2} + \cdots + 1$
- Let $c'(x) = a(x)b(x) \bmod (x^N - 1)$, then $c'(x) = c(x) \bmod \phi_N(x)$
- $\Rightarrow c_i = c'_i - c'_N$
  - Noise, i.e., $\|xe_y\|_\infty$ is "doubled";
  - Every coefficient is "lifted" by c'_N - creates dependency;
- ECC doesn't work on dependent errors.

"*Disadvantages*" II - Noise management

Rational: Use ECC to control errors

- Consider $c(x) = a(x)b(x) \bmod x^{N-1} + x^{N-2} + \cdots + 1$
- Let $c'(x) = a(x)b(x) \bmod (x^N - 1)$, then $c'(x) = c(x) \bmod \phi_N(x)$
- $\Rightarrow c_i = c'_i - c'_N$
    - Noise, i.e., $\|xe_y\|_\infty$ is "doubled";
    - Every coefficient is "lifted" by c'_N - creates dependency;
- ECC doesn't work on dependent errors.

Solution - ring switching

- multiply over $\phi_N(x)$, lift the final results to $x^N - 1$ ring;
- $c'(x) = c(x)(x-1)$
- Only use the coefficients of $1, x^2, x^4, x^6, \ldots$
- Security? $\mathbb{Z}[x]/\phi_N(x) \cong \mathbb{Z}[x]/(x^N - 1) \cap \{\text{Poly with root } 1\}$

Round2

The team

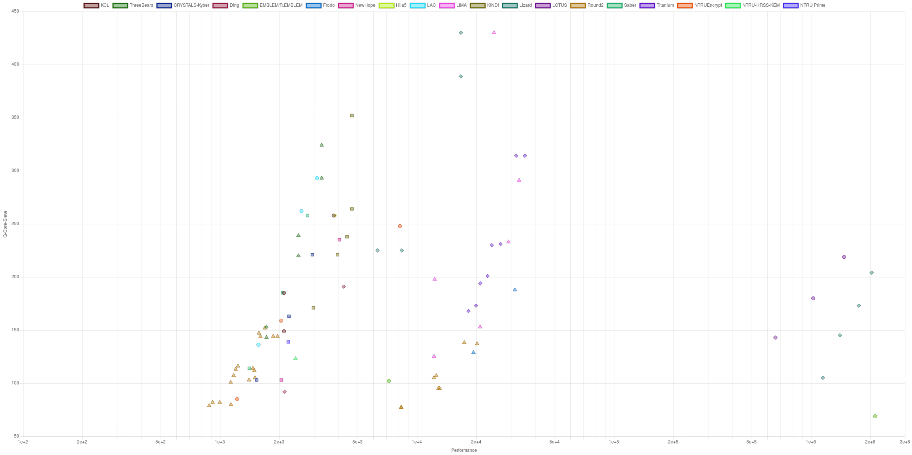
- Philips: Hayo Baan, Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Riemann, Ludo Tolhuizen, Jose Luis Torre Arce
- OnBoard Security: Zhenfei Zhang

Round5 = Round2 + HILA5's ECC

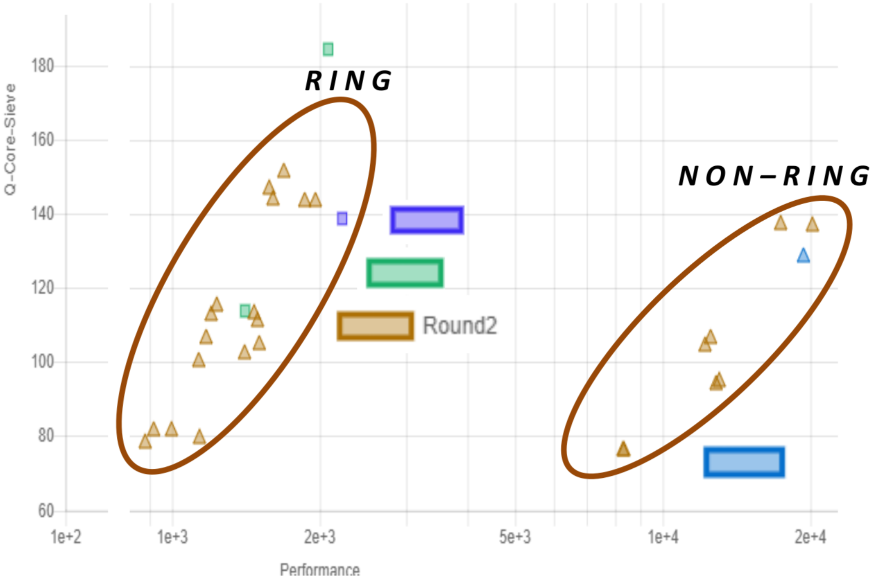
The team

- Philips: Hayo Baan, Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Riemann, Ludo Tolhuizen, Jose Luis Torre Arce
- Cisco: Scott Fluhrer
- Rambus: Mike Hamburg
- TU/e: Thijs Laarhoven
- PQShield: Markku-Juhani Olavi Saarinen
- OnBoard Security: Zhenfei Zhang

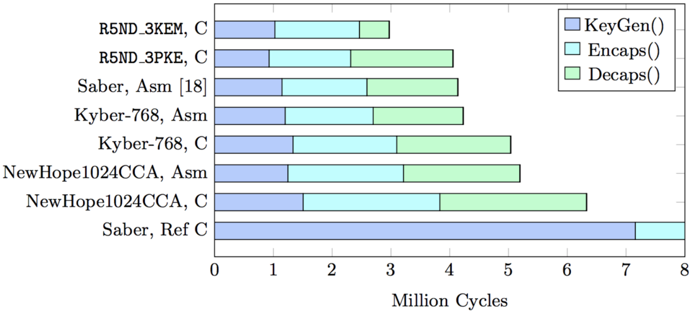
Performance



Performance



Performance



Deployment: TLS

```
Client                                          Server

ClientHello
ClientKeyShare            -------->
                          <--------      HelloRetryRequest

ClientHello
ClientKeyShare            -------->
                                              ServerHello
                                           ServerKeyShare
                                   {EncryptedExtensions*}
                                           {Certificate*}
                                   {CertificateRequest*+}
                                     {CertificateVerify*}
                          <--------            {Finished}
{Certificate*+}
{CertificateVerify*+}
{Finished}                -------->
[Application Data]        <------->      [Application Data]

      * message is not sent under some conditions
      + message is not sent unless client authentication
        is desired
```

Deployment: Hybrid solutions

```
INTERNET-DRAFT                                              W. Whyte
Intended Status: Experimental                    Security Innovation
Expires: 2017-XX-YY                                         Z. Zhang
                                                 Security Innovation
                                                          S. Fluhrer
                                                       Cisco Systems
                                                  O. Garcia-Morchon
                                                             Philips
                                                          2017-03-31


               Quantum-Safe Hybrid (QSH) Key Exchange
           for Transport Layer Security (TLS) version 1.3
                    draft-whyte-qsh-tls13-04.txt

  Client                                                  Server

  ClientHelloExtensions
  + qshDataExtension
    (QSHPKList)
  + qshNegotiateExtension
    (QSHSchemeIDList)        -------->
                                   HelloRetryRequestExtensions
                                       + qshNegotiateExtension
                             <-------- (AcceptQSHSchemeIDList)

  ClientHelloExtensions
  + qshDataExtension

    (QSHPKList)              -------->
                                          EncryptedExtensions*
                                            + qshDataExtension
                                               (QSHCipherList)
                             <--------              {Finished}
  {Finished}                 -------->

  ClassicSecret|QSHSecret    <-------> ClassicSecret|QSHSecret

        * previously known as SeverKeyShareExtensions
```

ARE WE DONE YET?

Another look at the security

$$
\begin{aligned}
\text{Round5} &\Rightarrow \text{Dec R-LWR over PC} \\
&\Rightarrow \text{Search R-LWE over PC} \\
&\Rightarrow \text{Search R-LWE over any ring} \\
&\Rightarrow \text{BDD over Ideal Lattices}
\end{aligned}
$$

Another look at the security

Round5 $\Rightarrow$ Dec R-LWR over PC

$\Rightarrow$ Search R-LWE over PC

$\Rightarrow$ Search R-LWE over any ring

$\Rightarrow$ BDD over Ideal Lattices

What about the hardness of Dec R-LWR?

Dec R-LWE as hard as search R-LWE

- Given $(a, b = as + e) \in \mathcal{R}^2$;
- Increase the first coefficient of b gradually and call Dec R-LWE oracle;
- Oracle will keep returning true till overflow - this tells us e_0;
- Repeat for all coefficients to learn e;
- Extract s from a noisy free sample $b' = as$.

What about the hardness of Dec R-LWR?

Dec R-LWE as hard as search R-LWE

- Given $(a, b = as + e) \in \mathcal{R}^2$;
- Increase the first coefficient of b gradually and call Dec R-LWE oracle;
- Oracle will keep returning true till overflow - this tells us e_0;
- Repeat for all coefficients to learn e;
- Extract s from a noisy free sample $b' = as$.

Dec R-LWR?

- We can't modify e - it is deterministic.

Our approach

Intuition

Search Problem $\geq$ Computational Problem $\geq$ Decisional Problem

- Computational Diffie-Hellman: given $\{g, g^x, g^y\}$, find g^{xy};
- Similarly, given $\{a, Round(as_1), Round(as_2)\}$, find $Round(as_1 s_2)$;
- This is the underlying problem for R-LWR-KEX
    - Dec R-LWR problem isn't essential.

The whole picture

Computational R-LWR $(a, b = Round(as))$

$\Rightarrow$ Computational Rounded R-LWE $(a, b = Round(as + e))$

$\Rightarrow$ Search R-LWE $(a, b = as + e)$

	R-LWE	R-LWR
Samples - KeyGen	2	1
Samples - Encrypt	3	1
Sampler	Gaussian	Uniform & Invertible
Modulus	$\Omega(n^{5.5} \log^{0.5} n)$	$\Omega(n^{3.75} \log^{0.25} n)$

Table: Performance comparison

If I have an unlimited fund

- Lattice based zero knowledge proofs;
- Lattice based group signatures;
- Sieving algorithms.


OBRIGADO
gracias
どうも
ARIGATO
hvala
DANK U
takk
MERSI
merci
謝謝
obrigado
danke schön
köszi
PALDIES
muchas gracias
ありがとう
TEŞEKKÜR EDERIM
MOLTE GRAZIE
GO RAIBH MAITH AGAT
danke
grazas
GRAZZI
THANK YOU
благодаря
TAK
どうも
THANKS
qujan
TAK
muchas gracias
vielen dank
PALDIES
DANK U
danke
OBRIGADO
mesi
감사합니다
köszi
благодаря
grazie
TACK
Gràcies
DZIEKI
TEŞEKKÜR EDERIM
muchas gracias
obrigado
спасибо
MULTUMESC
多謝
NA GODE

Backup materials: Lattice basics

Figure source: Wendy Cordero's High School Math Site

Lattice

Definition of a Lattice

- All the integral combinations of $d \leq n$ linearly independent vectors over $\mathbb{R}$

$$\mathcal{L} = \mathbb{Z}\,\boldsymbol{b}_1 + \cdots + \mathbb{Z}\,\boldsymbol{b}_d = \{\lambda_1 \boldsymbol{b}_1 + \cdots + \lambda_d \boldsymbol{b}_d \;:\; \lambda_i \in \mathbb{Z}\}$$

- d dimension.
- $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$ is a *basis*.

An example

$$\boldsymbol{B} = \begin{pmatrix} 5 & \frac{1}{2} & \sqrt{3} \\ \frac{3}{5} & \sqrt{2} & 1 \end{pmatrix}$$

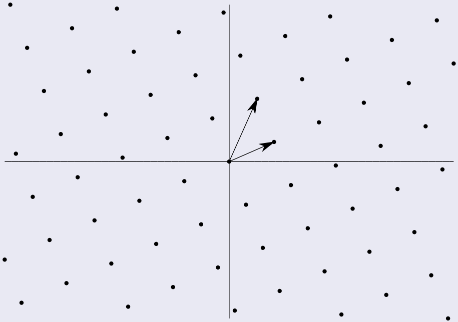
- $d = 2 \leq n = 3$
- In this talk, full rank integer Basis: $\boldsymbol{B} \in \mathbb{Z}^{n,n}$.

Example

A lattice $\mathcal{L}$

$$\boldsymbol{B} = \begin{pmatrix} 8 & 5 \\ 5 & 16 \end{pmatrix}$$

All lattice crypto talks start with an image of a dim-2 lattice

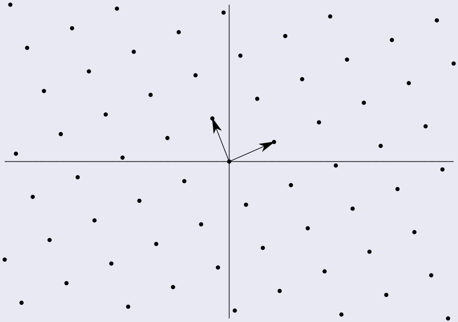


Example

A lattice $\mathcal{L}$

$$\boldsymbol{UB} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 8 & 5 \\ 5 & 16 \end{pmatrix} = \begin{pmatrix} 8 & 5 \\ -3 & 11 \end{pmatrix}$$

An infinity of basis

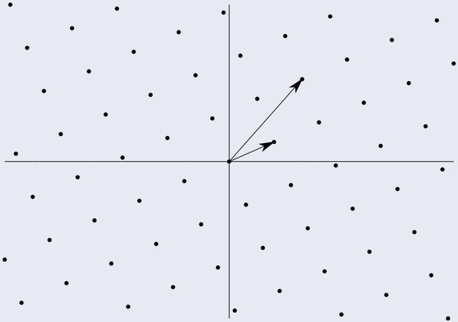


Example

A lattice $\mathcal{L}$

$$\boldsymbol{UB} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 8 & 5 \\ 5 & 16 \end{pmatrix} = \begin{pmatrix} 8 & 5 \\ 13 & 21 \end{pmatrix}$$

An infinity of basis

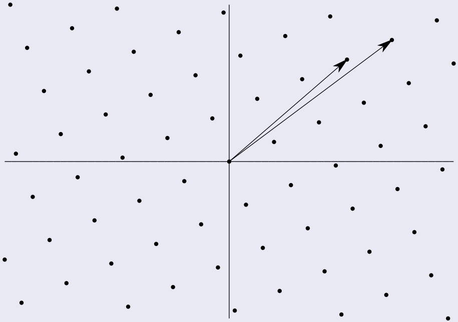


Example

A lattice $\mathcal{L}$

$$\boldsymbol{UB} = \begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 8 & 5 \\ 5 & 16 \end{pmatrix} = \begin{pmatrix} 29 & 31 \\ 21 & 26 \end{pmatrix}$$

An infinity of basis

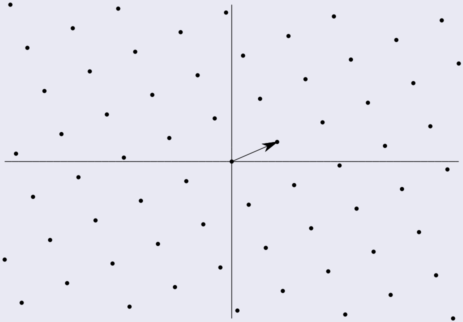


Example

The Shortest Vector and The First Minima

$$\boldsymbol{v} = \begin{pmatrix} 8 & 5 \end{pmatrix}, \text{ with } \lambda_1 = \sqrt{8^2 + 5^2} = 9.434$$

The Shortest Vector

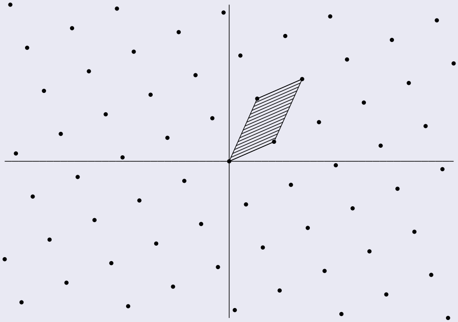


Example

The Determinant

$$\det \mathcal{L} = \sqrt{\det \left(\boldsymbol{B}\boldsymbol{B}^T \right)} = 103$$

The Fundamental Parallelepiped



NTRU lattice

NTRU ring

- Originally: $\mathbb{Z}_q[x]/(x^N - 1)$, q a power of 2, N a prime;
- Alternative 1: $\mathbb{Z}_q[x]/(x^N - x - 1)$, q a prime;
- Alternative 2: $\mathbb{Z}_q[x]/(x^N + 1)$, q a prime, N a power of 2

NTRU lattice

NTRU ring

- Originally: $\mathbb{Z}_q[x]/(x^N - 1)$, q a power of 2, N a prime;
- Alternative 1: $\mathbb{Z}_q[x]/(x^N - x - 1)$, q a prime;
- Alternative 2: $\mathbb{Z}_q[x]/(x^N + 1)$, q a prime, N a power of 2

Ring multiplications: $h(x) = f(x) \cdot g(x)$

- Compute $h'(x) = f(x) \times g(x)$ over $\mathbb{Z}[x]$
- Reduce $h'(x)$ mod $(x^N - 1)$ mod q

NTRU lattice

NTRU ring

- Originally: $\mathbb{Z}_q[x]/(x^N - 1)$, q a power of 2, N a prime;
- Alternative 1: $\mathbb{Z}_q[x]/(x^N - x - 1)$, q a prime;
- Alternative 2: $\mathbb{Z}_q[x]/(x^N + 1)$, q a prime, N a power of 2

Ring multiplications: $h(x) = f(x) \cdot g(x)$, alternatively

$$\langle h_0, \dots, h_{N-1} \rangle = \langle f_0, \dots, f_{N-1} \rangle \times \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{N-1} \\ g_{N-1} & g_0 & g_1 & \cdots & g_{N-2} \\ g_{N-2} & g_{N-1} & g_0 & \cdots & g_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_0 \end{bmatrix} \bmod q$$

NTRU lattice

NTRU assumption

- Decisional: given two small ring elements f and g; it is hard to distinguish $h = f/g$ from a uniformly random ring element;
- Computational: given h, find f and g.

NTRU lattice

NTRU assumption

- Decisional: given two small ring elements f and g; it is hard to distinguish $h = f/g$ from a uniformly random ring element;
- Computational: given h, find f and g.

NTRU lattice

$$\begin{bmatrix} qI_N & 0 \\ H & I_N \end{bmatrix} := \begin{bmatrix} q & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & q & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q & 0 & 0 & \dots & 0 \\ h_0 & h_1 & \dots & h_{N-1} & 1 & 0 & \dots & 0 \\ h_{N-1} & h_0 & \dots & h_{N-2} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \dots & h_0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

NTRU lattice

NTRU assumption

- Decisional: given two small ring elements f and g; it is hard to distinguish $h = f/g$ from a uniformly random ring element;
- Computational: given h, find f and g.

NTRU lattice $\mathcal{L} = \begin{bmatrix} qI_N & 0 \\ H & I_N \end{bmatrix}$

- $\langle g, f \rangle$ (and its cyclic rotations) are unique shortest vectors in $\mathcal{L}$;
- Decisional problem: decide if $\mathcal{L}$ has unique shortest vectors;
- Computational problem: find those vectors.
- Both are hard for random lattices.

NTRU lattice

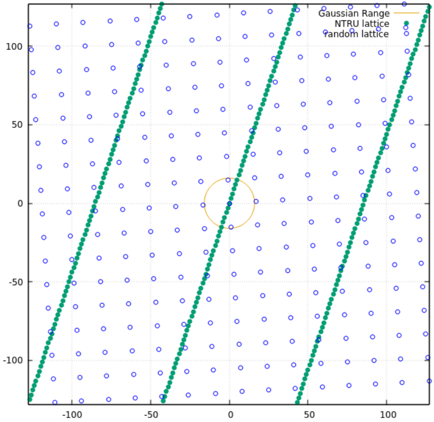
The real NTRU assumption

- NTRU lattice behaves the same as random lattices.

NTRU lattice $\mathcal{L} = \begin{bmatrix} qI_N & 0 \\ H & I_N \end{bmatrix}$

- $\langle g, f \rangle$ (and its cyclic rotations) are unique shortest vectors in $\mathcal{L}$;
- Decisional problem: decide if $\mathcal{L}$ has unique shortest vectors;
- Computational problem: find those vectors.
- Both are hard for random lattices.

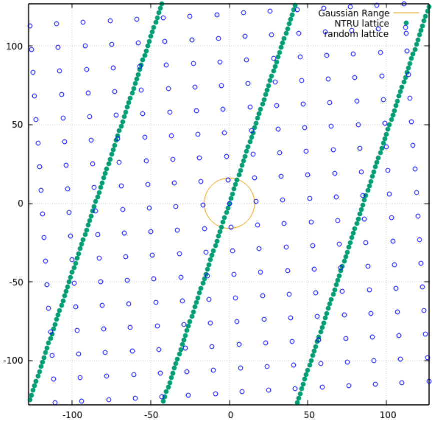
NTRU lattice vs random lattice



$$\begin{pmatrix} 256 & 0 \\ 172 & 1 \end{pmatrix} \qquad \begin{pmatrix} 256 & 0 \\ 17 & 1 \end{pmatrix}$$

$$(g, f) = (1, 3) \qquad v = (17, 1)$$

NTRU lattice vs random lattice



- Random lattice, SV $\approx$ Gaussian Heuristic length $= \sqrt{\frac{\dim}{2\pi e}} \det^{\frac{1}{\dim}}$
- NTRU lattice, unique shortest vectors $= \|g, f\|_2$

Lattice signatures

GGHSign	hash-then-sign	generic lattice
NTRUSign	hash-then-sign	NTRU lattice
Fiat Shamir with abort	FS, Rejection sampling	generic lattice
GPV	hash-then-sign	generic lattice
BLISS	FS, Rejection sampling	NTRU lattice
Dilithium	FS, Rejection sampling	generic lattice
Falcon	hash-then-sign	NTRU lattice
pqNTRUSign	HTS, Rejection sampling	NTRU lattice

GGHSign

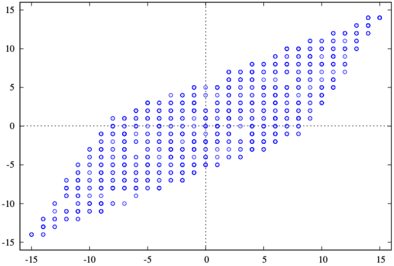
- Signing key: a good basis B
- Verification key a bad basis H
- Sign
  - Hash message to a vector $\mathbf{v}$
  - Use B to find the closest vector $\mathbf{c}$ (Babai's algorithm)
- Verification
  - Check $Dist(\mathbf{v} - \mathbf{c})$ is small

NTRUSign

- Good basis: (g,f)
- Bad basis: h

Transcript security

- Breaks GGHSign, NTRUSign;
- Each signature is a vector close to the lattice (info leakage);
- Recover enough of distance vectors (blue dots) gives away a good basis of the lattice;
- Seal the leakage with rejection sampling.



GPV sampler: a randomized Babai function

How it works

- A trapdoored lattice $\mathcal{L}$, i.e.

$$\mathcal{L}_A^{\perp} := \{v : Av = 0 \bmod q\}, \qquad \mathcal{L}_h := \{(u, v) : uh = v \bmod q\}$$

- A trapdoor S, or (g, f), and a smooth parameter $\eta_{\varepsilon}(\mathcal{L})$
- A target lattice point $\mathbf{v}$
- Outputs another vector $\mathbf{s}$, s.t.
  - $\mathbf{s}$ is uniform over $\mathcal{L}$
  - $dist(\mathbf{s}, \mathbf{v})$ Gaussian over $\mathbb{Z}^n$

Bottle neck: trapdoor generation

- Bonsai Tree, Gadget matrix, NTRU lattices ...

Falcon

- Falcon = GPV + NTRUSign + more ticks

Falcon

Pierre-Alain Fouque[1] Jeffrey Hoffstein[2] Paul Kirchner[1] Vadim Lyubashevsky[3] Thomas Pornin[4]
Thomas Prest[5] Thomas Ricosset[5] Gregor Seiler[3] William Whyte[6] Zhenfei Zhang[6]

Falcon in a Nutshell

We work over the cyclotomic ring $\mathcal{R} = \mathbb{Z}_q[x]/(x^n + 1)$.

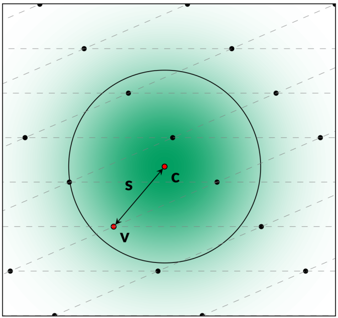
- **Keygen()**
  1. Generate matrices **A**, **B** with coefficients in $\mathcal{R}$ such that
     - $\mathbf{BA} = 0$
     - **B** has small coefficients
  2. pk $\leftarrow$ **A**
  3. sk $\leftarrow$ **B**

- **Sign(m,sk)**
  1. Compute **c** such that $\mathbf{cA} = H(\mathrm{m})$
  2. $\mathbf{v} \leftarrow$ "a vector in the lattice $\Lambda(\mathbf{B})$, close to **c**"
  3. $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$

  The signature sig is $\mathbf{s} = (s_1, s_2)$

- **Verify(m,pk sig)**
  Accept iff:
  1. **s** is short
  2. $\mathbf{sA} = H(\mathrm{m})$

$\Rightarrow$



Performance comparison

	A	B	I	J	K	L
1		Category	sk	pk	bytes	
2	Dilithium_medium	Lattices	2,800	1,184	2,044	
3	Dilithium_recommended	Lattices	3,504	1,472	2,701	
4	Dilithium_very_high	Lattices	3,856	1,760	3,366	
5						
6	falcon1024	Lattices	8,193	1,793	1,330	
7	falcon512	Lattices	4,097	897	690	
8	falcon768	Lattices	6,145	1,441	1,077	
9						
10	qTesla_128	Lattices	2,112	4,128	3,104	
11	qTesla_192	Lattices	8,256	8,224	6,176	
12	qTesla_256	Lattices	8,256	8,224	6,176	
13						
14	Gaussian-1024	Lattices	2,604	2,065	2,065	
15						
16						

Raptor

- Raptor = Falcon + anonymity (stealth mode)

Raptor

The scheme

- **Setup** Output a hash function $\mathcal{H} : \{*\} \rightarrow D_b$ and a random $\mathbf{h}$.
- **KeyGen** Return $pk = \mathbf{a} := \mathbf{g}/\mathbf{f}$ and $sk = (\mathbf{f}, \mathbf{g})$.
- **Signing**
  - Input $\{pk_1, \ldots, pk_\ell\}$, sk_π and μ;
  - For $i \in [1, \cdots, \ell]$ and $i \neq \pi$, pick $\mathbf{m}_i$ and $\mathbf{r}_i$. Compute $\mathbf{c}_i = \mathbf{h}\mathbf{m}_i + \mathbf{a}_i\mathbf{r}_i$.
  - For $i = \pi$, pick $\mathbf{c}_\pi$.
  - Compute $\mathbf{m}_\pi$ such that

$$\mathbf{m}_1 \oplus \cdots \oplus \mathbf{m}_\pi \oplus \cdots \oplus \mathbf{m}_\ell = \mathcal{H}(\mu, \mathbf{c}_1, \cdots, \mathbf{c}_\ell, pk_1, \ldots, pk_l). \qquad (1)$$

  - Set $\mathbf{u} = \mathbf{c}_\pi - \mathbf{h}\mathbf{m}_\pi$
  - Set $\mathbf{r}_\pi = \text{Falcon.sign}(\mathbf{u}, sk_\pi)$
  - Signature $(\mathbf{r}_1, \ldots, \mathbf{r}_\pi, \mathbf{m}_i, \ldots, \mathbf{m}_\pi)$
- **Verify**
  - For $i \in [1, \cdots, \ell]$, generate $\mathbf{c}_i = \mathbf{h}\mathbf{m}_i + \mathbf{a}_i\mathbf{r}_i$
  - Check Equation (1).

Raptor

Security

- Public key security: as hard as breaking NTRU
- Strong Unforgeability: as hard as forging a Falcon signature
- Strong Anonymity: $(\mathbf{m}_\pi, \mathbf{r}_\pi)$ statistically IND from $(\mathbf{m}_i, \mathbf{r}_i)$

Linkable Raptor

- Use one time signature to generate a tag for each pk
- Tag is enforced in verification
- Same tag = linked.

Wrap up

Features

- First lattice based linkable ring signature
- Do not require NIZK
- Competitive performance to classical solutions
- 50 to 100x smaller than known (none-implementable) PQC solutions