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 Prime

Daniel J. Bernstein, Chitchanok Chuengsatiansup,
Tanja Lange, and Christine van Vredendaal

Technische Universiteit Eindhoven

20 January 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 (p, q, t), p prime, integers t, q, $\gcd(p, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^p - 1)$.

NTRU

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

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

  where H corresponds to multiplication by $h/3$ modulo $x^p - 1$.
- (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 (p, q, t), p prime, integers t, q, $\gcd(p, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^p - 1)$, some use additional reduction modulo q, ring denoted by R_q.

Classic NTRU

- System parameters (p, q, t), p prime, integers t, q, $\gcd(p, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^p - 1)$, 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 (p, q, t), p prime, integers t, q, $\gcd(p, q) = 1$.
- All computations done in ring $R = \mathbf{Z}[x]/(x^p - 1)$, 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.

Maps on R_q

- Evaluation at 1 attack: $c(1) = m(1)$
- Consider $R_q = (\mathbf{Z}/q)[x]/(x^p - 1)$.
- Can possibly get more information on m from homomorphism $\psi : R_q \to T$, for some ring T.
- Attacker applies map to $h = 3g/f$ and to $c = m + hr$ in R_q.
- Typical NTRU choice: $q = 2048$ leads to natural ring maps from $(\mathbf{Z}/2048)[x]/(x^p - 1)$ to
  - $(\mathbf{Z}/2)[x]/(x^p - 1)$,
  - $(\mathbf{Z}/4)[x]/(x^p - 1)$,
  - $(\mathbf{Z}/8)[x]/(x^p - 1)$, etc.
- Unclear whether these can be exploited to get information on m.
- 2004 Smart–Vercauteren–Silverman: Maybe. Complicated.
- Typical R-LWE case: take $(\mathbf{Z}/q)[x]/(x^n + 1)$ with n power of 2 so that $x^n + 1$ splits completely modulo q.

Do these maps damage security?

Unclear.

Do these maps damage security?

Consider generalized setting $(\mathbf{Z}/q)[x]/P$ for some polynomial P.
Construct bad cases of P and q, break those systems:

- 2014 Eisenträger–Hallgren–Lauter,
- 2015 Elias–Lauter–Ozman–Stange,
- 2016 Chen–Lauter–Stange.

Recent Castryck–Iliashenko–Vercauteren cryptanalysis of $(\mathbf{Z}/q)[x]/P$ covers Elias–Lauter–Ozman–Stange cases without dependence on q, but not more recent Chen–Lauter–Stange ones.

Some polynomials P are bad because they lead to very low noise in some coordinates independent of q.

But for some pairs P, q the properties of P modulo q matter.
(Yeah, number theory!)

Do these maps damage security?

Consider generalized setting $(\mathbf{Z}/q)[x]/P$ for some polynomial P.
Construct bad cases of P and q, break those systems:

- 2014 Eisenträger–Hallgren–Lauter,
- 2015 Elias–Lauter–Ozman–Stange,
- 2016 Chen–Lauter–Stange.

Recent Castryck–Iliashenko–Vercauteren cryptanalysis of $(\mathbf{Z}/q)[x]/P$ covers Elias–Lauter–Ozman–Stange cases without dependence on q, but not more recent Chen–Lauter–Stange ones.

Some polynomials P are bad because they lead to very low noise in some coordinates independent of q.

But for some pairs P, q the properties of P modulo q matter.
(Yeah, number theory!)
Latest addition:
attack on multiquadratic fields (Christine's and Dan's talk yesterday).

NTRU Prime

Born out of paranoia, aka. risk management.

- Talk at Oberwolfach 2013 by Dan with rough proposal.
- Feb 2014: more detailed blogpost by Dan `https://blog.cr.yp.to/20140213-ideal.html` focussing on avenues for attacks.
- Subfield-logarithm attack strategy, sometimes much faster than Gentry–Szydlo.

NTRU Prime

Born out of paranoia, aka. risk management.

- Talk at Oberwolfach 2013 by Dan with rough proposal.
- Feb 2014: more detailed blogpost by Dan `https://blog.cr.yp.to/20140213-ideal.html` focussing on avenues for attacks.
- Subfield-logarithm attack strategy, sometimes much faster than Gentry–Szydlo.
- Now fully worked out NTRU Prime and Streamlined NTRU Prime (with parameters and implementation).
- NTRU Prime
  - avoids large proper subfields;
  - avoids ring homomorphisms to smaller rings;
  - avoids an easy to find fundamental basis of short units which is useful in Soliloquy attack (Campbell–Groves–Shepherd) and Cramer–Ducas–Peikert–Regev.

NTRU Prime ring

- Differences with NTRU:
  prime degree, large Galois group, inert modulus.

NTRU Prime ring

- Differences with NTRU:
  prime degree, large Galois group, inert modulus.
- Choose monic irreducible polynomial $P \in \mathbf{Z}[x]$.
- Choose prime q such that P is irreducible modulo q; this means that q is inert in $\mathcal{R} = \mathbf{Z}[x]/P$ and $(\mathbf{Z}/q)[x]/P$ is a field.

NTRU Prime ring

- Differences with NTRU:
  prime degree, large Galois group, inert modulus.
- Choose monic irreducible polynomial $P \in \mathbf{Z}[x]$.
- Choose prime q such that P is irreducible modulo q; this means that q is inert in $\mathcal{R} = \mathbf{Z}[x]/P$ and $(\mathbf{Z}/q)[x]/P$ is a field.
- Further choose P of prime degree p with large Galois group.
- Specifically, set $P = x^p - x - 1$. This has Galois group S_p of size $p!$.
- Streamlined NTRU Prime works over the NTRU Prime *field*

$$\mathcal{R}/q = (\mathbf{Z}/q)[x]/(x^p - x - 1).$$

NTRU Prime: added defenses

Prime degree, large Galois group, inert modulus.

NTRU Prime: added defenses

Prime degree, large Galois group, inert modulus.

- ➜ Only subfields of $\mathbf{Q}[x]/P$ are itself and $\mathbf{Q}$. Avoids structures used by, e.g., multiquad attack.
- ➜ Large Galois group means no easy to compute automorphisms. Roots of P live in degree-$p!$ extension. Avoids structures used by Campbell–Groves–Shepherd attack (obtaining short unit basis). No hopping between units, so no easy way to extend from some small unit to a fundamental system of short units.
- ➜ No ring homomorphism to smaller nonzero rings. Avoids structures used by Chen–Lauter–Stange attack.

NTRU Prime: added defenses

Prime degree, large Galois group, inert modulus.

- Only subfields of $\mathbf{Q}[x]/P$ are itself and $\mathbf{Q}$. Avoids structures used by, e.g., multiquad attack.
- Large Galois group means no easy to compute automorphisms. Roots of P live in degree-$p!$ extension. Avoids structures used by Campbell–Groves–Shepherd attack (obtaining short unit basis). No hopping between units, so no easy way to extend from some small unit to a fundamental system of short units.
- No ring homomorphism to smaller nonzero rings. Avoids structures used by Chen–Lauter–Stange attack.

Irreducibility also avoids the evaluation-at-1 attack which simplifies padding.

Streamlined NTRU Prime: private and public key

- System parameters (p, q, t), p, q prime, $q \geq 32t + 1$.
- Pick g small in $\mathcal{R}$

$$g = g_0 + \cdots + g_{p-1}x^{p-1} \text{ with } g_i \in \{-1, 0, 1\}$$

  No weight restriction on g, only size restriction on coefficients; g required to be invertible in $\mathcal{R}/3$.

- Pick t-small $f \in \mathcal{R}$

$$f = f_0 + \cdots + f_{p-1}x^{p-1} \text{ with } f_i \in \{-1, 0, 1\} \text{ and } \sum |f_i| = 2t$$

  Since $\mathcal{R}/q$ is a field, f is invertible.

- Compute public key $h = g/(3f)$ in $\mathcal{R}/q$.
- Private key is f and $1/g \in \mathcal{R}/3$.

Streamlined NTRU Prime: private and public key

- System parameters (p, q, t), p, q prime, $q \geq 32t + 1$.
- Pick g small in $\mathcal{R}$

$$g = g_0 + \cdots + g_{p-1}x^{p-1} \text{ with } g_i \in \{-1, 0, 1\}$$

  No weight restriction on g, only size restriction on coefficients; g required to be invertible in $\mathcal{R}/3$.
- Pick t-small $f \in \mathcal{R}$

$$f = f_0 + \cdots + f_{p-1}x^{p-1} \text{ with } f_i \in \{-1, 0, 1\} \text{ and } \sum |f_i| = 2t$$

  Since $\mathcal{R}/q$ is a field, f is invertible.
- Compute public key $h = g/(3f)$ in $\mathcal{R}/q$.
- Private key is f and $1/g \in \mathcal{R}/3$.
- Difference with NTRU: more key options, 3 in denominator.

Streamlined NTRU Prime: KEM/DEM

- Streamlined NTRU Prime is a Key Encapsulation Mechanism (KEM).
- Combine with Data Encapsulation Mechanism (DEM) to send messages. (Fancy name for symmetric authenticated encryption under shared key.)

Streamlined NTRU Prime: KEM/DEM

- Streamlined NTRU Prime is a Key Encapsulation Mechanism (KEM).
- Combine with Data Encapsulation Mechanism (DEM) to send messages. (Fancy name for symmetric authenticated encryption under shared key.)

KEM:

- Alice looks up Bob's public key h.
- Picks t-small $r \in \mathcal{R}$ (i.e., $r_i \in \{-1, 0, 1\}, \sum |r_i| = 2t$).
- Computes hr in $\mathcal{R}/q$, lifts coefficients to $\mathbf{Z} \cap [-(q-1)/2, (q-1)/2]$.

Streamlined NTRU Prime: KEM/DEM

- Streamlined NTRU Prime is a Key Encapsulation Mechanism (KEM).
- Combine with Data Encapsulation Mechanism (DEM) to send messages. (Fancy name for symmetric authenticated encryption under shared key.)

KEM:

- Alice looks up Bob's public key h.
- Picks t-small $r \in \mathcal{R}$ (i.e., $r_i \in \{-1, 0, 1\}, \sum |r_i| = 2t$).
- Computes hr in $\mathcal{R}/q$, lifts coefficients to $\mathbf{Z} \cap [-(q-1)/2, (q-1)/2]$.
- Rounds each coefficient to the nearest multiple of 3 to get c.
- Computes $\text{hash}(r) = (C|K)$.
- Sends $(C|c)$, uses session key K for DEM.

Rounding hr saves bandwidth and adds same entropy as adding ternary m.

Streamlined NTRU Prime: decapsulation

Bob decrypts $(C|c)$:

- Reminder $h = g/(3f)$ in $\mathcal{R}/q$.
- Computes $3fc = 3f(hr + m) = gr + 3fm$ in $\mathcal{R}/q$, lifts coefficients to $\mathbf{Z} \cap [-(q-1)/2, (q-1)/2]$.
- Reduces the coefficients modulo 3 to get $a = gr \in \mathcal{R}/3$.
- Computes $r' = a/g \in \mathcal{R}/3$, lifts r' to $\mathcal{R}$.
- Computes $\mathrm{hash}(r') = (C'|K')$ and c' as rounding of hr'.
- Verifies that $c' = c$ and $C' = C$.

If all checks verify, $K = K'$ is the session key between Alice and Bob and can be used in a data encapsulation mechanism (DEM).

Choosing $q \geq 32t + 1$ means no decryption failures, so $r = r'$ and verification works unless $(C|c)$ was incorrectly generated or tempered with.

Streamlined NTRU Prime Security

- Short recap:

	NTRU	**R-LWE**	**NTRU Prime**
Polynomial P	$x^p - 1$	$x^p + 1$	$x^p - x - 1$
Degree p	prime	power of 2	prime
Modulus q	2^d	prime	prime
$\#$ factors of P in $\mathcal{R}/q$	> 1	p	1
$\#$ proper subfields	> 1	many	1
Every m encryptable	✗	✓	✓
No decryption failures	✗	✗	✓

Streamlined NTRU Prime Security

- Short recap:

	NTRU	**R-LWE**	**NTRU Prime**
Polynomial P	$x^p - 1$	$x^p + 1$	$x^p - x - 1$
Degree p	prime	power of 2	prime
Modulus q	2^d	prime	prime
$\#$ factors of P in $\mathcal{R}/q$	> 1	p	1
$\#$ proper subfields	> 1	many	1
Every m encryptable	✗	✓	✓
No decryption failures	✗	✗	✓

- Because of the last 2 ✓'s the analysis is simpler than that of NTRU.

NTRU Prime Security: parameters

- We investigated security against the strongest known attacks; meet-in-the-middle (mitm), hybrid attack of BKZ and mitm, and lattice sieving.

p	q	t	Key size	Ciphertext Size	Security
739	9829	246	9.9 Kb	9.1 Kb	232
761	4591	143	9.2 Kb	8.1 Kb	248

- We underestimated cost of hybrid attack, see Thomas' talk on Thursday.
- Security is given as classical security. Quantum computers will speed up by less than squareroot.

NTRU Prime Security: parameters

- We investigated security against the strongest known attacks; meet-in-the-middle (mitm), hybrid attack of BKZ and mitm, and lattice sieving.

p	q	t	Key size	Ciphertext Size	Security
739	9829	246	9.9 Kb	9.1 Kb	232
761	4591	143	9.2 Kb	8.1 Kb	248

- We underestimated cost of hybrid attack, see Thomas' talk on Thursday.
- Security is given as classical security. Quantum computers will speed up by less than squareroot.

But, is it still fast?

Polynomial Multiplication

- Main bottleneck is polynomial multiplication
- Classic choices of $x^p - 1$ and $x^n + 1$ have very fast reduction.
- NTRU uses $x^p - 1$ for p prime and $q = 2^N$.
- Most R-LWE systems use $x^n + 1$, with $n = 2^t$; q prime.
  Typical implementations use the number-theoretic transform (NTT).
  This requires q to be "NTT-friendly", i.e., $x^n + 1$ splits into linear factors modulo q, so $q \equiv 1 \bmod 2n$;
  e.g. $n = 1024$ and $q = 6 \cdot 2048 + 1$.
- Complete factorization of $x^n + 1$ modulo q is also used in search-to-decision problem reductions.
- Obvious benefit: NTT is fast.
- Not so obvious downside: NTT friendly combinations are rare – likely to overshoot security targets in some direction.

Multiplication for NTRU Prime

- How to compute efficiently in $\mathbf{Z}[x]/(x^p - x - 1)$?
- Reduction is not too bad, but no special tricks for multiplication.
- Multiplication algorithms considered:
  - Toom (3–7),
  - refined Karatsuba,
  - arbitrary degree variant of Karatsuba (3–7 levels).

Multiplication for NTRU Prime

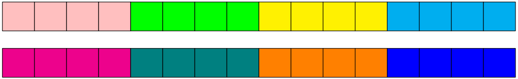
- How to compute efficiently in $\mathbf{Z}[x]/(x^p - x - 1)$?
- Reduction is not too bad, but no special tricks for multiplication.
- Multiplication algorithms considered:
  - Toom (3–7),
  - refined Karatsuba,
  - arbitrary degree variant of Karatsuba (3–7 levels).
- Best operation count found so far for 768×768:
  - 5-level refined Karatsuba up to 128×128, combined with
  - Toom6: evaluated at $0, \pm 1, \pm 2, \pm 3, \pm 4, 5, \infty$.

  Toom reconstructs a polynomial based on evaluation. We group coefficients into 6 chunks of size 128 and use Karatsuba for multiplying these smaller chunks.

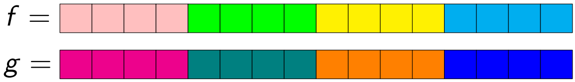
Vectorization

$f =$

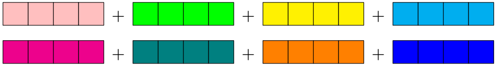
$g =$



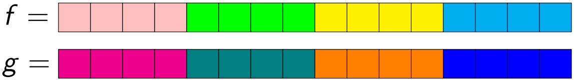
Vectorization



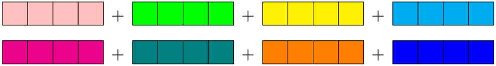
- Toom & Karatsuba
  - cut polynomials into smaller parts; independent operations on the parts



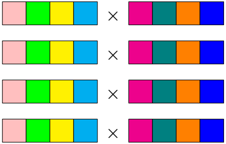
Vectorization



- Toom & Karatsuba
  - cut polynomials into smaller parts; independent operations on the parts



- Vectorization
  - vectorize *across* independent multiplications



Performance

- Theoretical lower bound
  - 0.125 cycles per floating-point operation.
  - Permutations fully interleavable. First parameter set.

	mul	con mult	add	shift	total
op.	42768	9700	98548	6385	157401
cycles	5346	1213	12319	799	19677

- Current implementation
  - Benchmarked performance: 51488 cycles
  - possibly due to dependency, latency, scheduling issues.

Performance

- Theoretical lower bound
  - 0.125 cycles per floating-point operation.
  - Permutations fully interleavable. First parameter set.

	mul	con mult	add	shift	total
op.	42768	9700	98548	6385	157401
cycles	5346	1213	12319	799	19677

- Current implementation
  - Benchmarked performance: 51488 cycles
  - possibly due to dependency, latency, scheduling issues.
  - R-LWE with 40000 cycles using NTT in New Hope paper by Alkim, Ducas, Pöppelmann, and Schwabe.
    Now even faster implementation in Microsoft Research's Lattice Cryptography Library.
  - For NTRU Prime, further optimization in progress.
  - This level of paranoia is not too expensive (compared with unstructured LWE or Goppa-code McEliece).

Bonus slides on attacks

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

- Christine's talk gives full explanation and new memory reduction.
- Idea: split the possibilities for f in two parts

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

- 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

- Christine's talk gives full explanation and new memory reduction.
- Idea: split the possibilities for f in two parts

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

- 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

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

  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.

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.

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.
- In NTRU Prime $P = x^p - x - 1$, so reduction modulo P changes density and weight, e.g.

$$(x^4 - x^2 + 1) \cdot x \equiv (x + 1) - x^3 + x = x^3 + 2x + 1 \bmod (x^5 - x - 1).$$

- For small i up to $p - 1 - \deg(f)$ have shifted (valid) target.
- Very unlikely that any $x^i f$ for large i produces viable keys; first reduction occurs on average at $i = p/(2t)$.

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

- Number of choices for f is

$$\binom{p}{2t} 2^{2t}$$

because f is t-small, signs of those $2t$ coefficients are random.

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

- Number of choices for f is

$$\binom{p}{2t}2^{2t}$$

because f is t-small, signs of those $2t$ coefficients are random.

- We (over-)estimate number of viable rotations by $p - t$.
- Running time / memory mitm against Streamlined NTRU Prime

$$L = \frac{\sqrt{\binom{p}{2t}2^{2t}}}{\sqrt{2(p-t)}}.$$

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

- Number of choices for f is

$$\binom{p}{2t}2^{2t}$$

because f is t-small, signs of those $2t$ coefficients are random.
- We (over-)estimate number of viable rotations by $p - t$.
- Running time / memory mitm against Streamlined NTRU Prime

$$L = \frac{\sqrt{\binom{p}{2t}2^{2t}}}{\sqrt{2(p-t)}}.$$

- Memory requirement can be reduced.

Security against lattice sieving

Lattice attack setup is same as for NTRU.

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

$$\begin{pmatrix} k & f \end{pmatrix} \begin{pmatrix} qI & 0 \\ H & I \end{pmatrix} = \begin{pmatrix} g & f \end{pmatrix}.$$

Security against lattice sieving

Lattice attack setup is same as for NTRU.

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

$$\begin{pmatrix} k & f \end{pmatrix} \begin{pmatrix} qI & 0 \\ H & I \end{pmatrix} = \begin{pmatrix} g & f \end{pmatrix}.$$

- 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 Streamlined NTRU Prime 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 Streamlined NTRU Prime lattice, then perform mitm on the rest.
- Use BKZ on submatrix B to get B':

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

- 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].

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].
- For detailed formulas and justifications, see our paper `https://eprint.iacr.org/2016/461`.

Bonus slides: why automorphisms matter

Targets and history:

- 2014.10 Campbell–Groves–Shepherd describe an ideal-lattice-based system "Soliloquy"; claim quantum poly-time key recovery.
- 2010 Smart–Vercauteren system is practically identical to Soliloquy.
- 2009 Gentry system (simpler version described at STOC) has the same key-recovery problem.
- 2012 Garg–Gentry–Halevi multilinear maps have the same key-recovery problem (and many other security issues).

Smart–Vercauteren; Soliloquy

- Parameter: $k \geq 1$.
- Define $R = \mathbf{Z}[x]/\Phi_{2^k}$.
- Public key: prime q and $c \in \mathbf{Z}/q$.
- Secret key: short element $g \in R$ with $gR = qR + (x - c)R$; i.e., short generator of the ideal $qR + (x - c)R$.

Smart–Vercauteren; Soliloquy

- Parameter: $k \geq 1$.
- Define $R = \mathbf{Z}[x]/\Phi_{2^k}$.
- Public key: prime q and $c \in \mathbf{Z}/q$.
- Secret key: short element $g \in R$ with $gR = qR + (x - c)R$; i.e., short generator of the ideal $qR + (x - c)R$.
- 1993 Cohen textbook "A course in computational algebraic number theory" explains how to find generators.

Smart–Vercauteren; Soliloquy

- Parameter: $k \geq 1$.
- Define $R = \mathbf{Z}[x]/\Phi_{2^k}$.
- Public key: prime q and $c \in \mathbf{Z}/q$.
- Secret key: short element $g \in R$ with $gR = qR + (x - c)R$; i.e., short generator of the ideal $qR + (x - c)R$.
- 1993 Cohen textbook "A course in computational algebraic number theory" explains how to find generators.
- Smart–Vercauteren comment that this would take exponential time.
- But it actually takes subexponential time. Same basic idea as NFS.
- Campbell–Groves–Shepherd claim quantum poly time. Claim disputed by Biasse, not defended by CGS.

Smart–Vercauteren; Soliloquy

- Parameter: $k \geq 1$.
- Define $R = \mathbf{Z}[x]/\Phi_{2^k}$.
- Public key: prime q and $c \in \mathbf{Z}/q$.
- Secret key: short element $g \in R$ with $gR = qR + (x - c)R$; i.e., short generator of the ideal $qR + (x - c)R$.
- 1993 Cohen textbook "A course in computational algebraic number theory" explains how to find generators.
- Smart–Vercauteren comment that this would take exponential time.
- But it actually takes subexponential time. Same basic idea as NFS.
- Campbell–Groves–Shepherd claim quantum poly time. Claim disputed by Biasse, not defended by CGS.
- 2016 Biasse–Song: different algorithm that takes quantum poly time, building on 2014 Eisenträger–Hallgren–Kitaev–Song.

How to get a short generator?

- Have ideal I of R.
- Want short g with $gR = I$; have g' with $g'R = I$.
- Know $g' = ug$ for some unit $u \in R^*$.
- To find u move to log lattice.

$$\mathrm{Log}\, g' = \mathrm{Log}\, u + \mathrm{Log}\, g,$$

  where Log is Dirichlet's log map.
- Dirichlet's unit theorem:
  $\mathrm{Log}\, R^*$ is a lattice of known dimension.
- Finding $\mathrm{Log}\, u$ is a closest-vector problem in this lattice.

Quote from Campbell–Groves–Shepherd

"A simple generating set for the cyclotomic units is of course known. The image of $\mathcal{O}^\times$ [here R^*] under the logarithm map forms a lattice. The determinant of this lattice turns out to be much bigger than the typical loglength of a private key α [here g], so it is easy to recover the causally short private key given *any* generator of $\alpha\mathcal{O}$ [here I], *e.g.* via the LLL lattice reduction algorithm."

Automorphisms

- $x \mapsto x^3$, $x \mapsto x^5$, $x \mapsto x^7$, etc. are automorphisms of $R = \mathbf{Z}[x]/\Phi_{2^k}$.
- Easy to see $(1 - x^3)/(1 - x) \in R^*$; for inverse use expansion.
- "Cyclotomic units" are defined as

$$R^* \cap \left\{ \pm x^{e_0} \prod_i (1 - x^i)^{e_i} \right\}.$$

- Weber's conjecture:

  All elements of R^* are cyclotomic units.
- Experiments confirm that SV is quickly broken by LLL using, e.g., 1997 Washington textbook basis for cyclotomic units.
- Shortness of basis is critical; this was not highlighted in CGS analysis.