NTRU Prime

Daniel J. Bernstein

University of Illinois at Chicago &
Technische Universiteit Eindhoven

---

cr.yp.to/papers.html
#ntruprime is joint work with:

Chitchanok Chuengsatiansup
Tanja Lange
Christine van Vredendaal

Technische Universiteit Eindhoven

Focus of this talk: motivation.

Can we predict future attacks?

1996 Dobbertin–Bosselaers–Preneel "RIPEMD-160: a strengthened version of RIPEMD": "It is anticipated that these techniques can be used to produce collisions for MD5 and perhaps also for RIPEMD. This will probably require an additional effort, but it no longer seems as far away as it was a year ago."

1996 Robshaw: Collisions "should be expected"; upgrade "when practical and convenient".

Imagine someone responding: "This is completely out of line. The attack by Dobbertin does *not* break any normal usage of MD5, so what exactly is the point of preventing it? This speculation about MD5 collisions is controversial and non-scientific, and creates confusion on the state of the art. Recommending alternative hash functions is at the very least quite premature."

Imagine someone responding: "This is completely out of line. The attack by Dobbertin does *not* break any normal usage of MD5, so what exactly is the point of preventing it? This speculation about MD5 collisions is controversial and non-scientific, and creates confusion on the state of the art. Recommending alternative hash functions is at the very least quite premature."

Clearly not a real cryptographer. Maybe a standards organization.

Now imagine a religious fanatic saying that all of these functions are worse than "provably secure" cryptographic hash functions.

Now imagine a religious fanatic saying that all of these functions are worse than "provably secure" cryptographic hash functions.

1991 "provably secure" example, Chaum–van Heijst–Pfitzmann:

Choose p sensibly.
Define $C(x, y) = 4^x 9^y \bmod p$
for suitable ranges of x and y.

Simple, beautiful, structured.
Very easy security reduction:
finding C collision implies
computing a discrete logarithm.

CvHP is very bad cryptography.
Horrible security for its speed.
Far worse security record than
standard "unstructured"
compression-function designs.

Security losses in C include
1922 Kraitchik (index calculus);
1986 Coppersmith–Odlyzko–
Schroeppel (NFS predecessor);
1993 Gordon (general DL NFS);
1993 Schirokauer (faster NFS);
1994 Shor (quantum poly time).

CvHP is very bad cryptography.
Horrible security for its speed.
Far worse security record than
standard "unstructured"
compression-function designs.

Security losses in C include
1922 Kraitchik (index calculus);
1986 Coppersmith–Odlyzko–
Schroeppel (NFS predecessor);
1993 Gordon (general DL NFS);
1993 Schirokauer (faster NFS);
1994 Shor (quantum poly time).

Imagine someone in 1991 saying
"DL security is well understood".

We still use discrete logs for *pre-quantum public-key* crypto.

Which DL groups are best?

We still use discrete logs for *pre-quantum public-key* crypto. Which DL groups are best?

1986 Miller proposes ECC. Gives detailed arguments that index calculus "is not likely to work on elliptic curves."

We still use discrete logs for *pre-quantum public-key* crypto. Which DL groups are best?

1986 Miller proposes ECC. Gives detailed arguments that index calculus "is not likely to work on elliptic curves."

1997 Rivest: "Over time, this may change, but for now trying to get an evaluation of the security of an elliptic-curve cryptosystem is a bit like trying to get an evaluation of some recently discovered Chaldean poetry."

Are RSA, DSA, etc. less scary?
These systems have structure
enabling attacks such as NFS.
Many optimization avenues.
Attacks keep getting better.
$>$100 scientific papers.
Still many unexplored avenues.

How many people understand
the state of the art?

Are RSA, DSA, etc. less scary?
These systems have structure
enabling attacks such as NFS.
Many optimization avenues.
Attacks keep getting better.
$>$100 scientific papers.
Still many unexplored avenues.

How many people understand
the state of the art?

Recurring themes in attacks:
factorizations of ring elements;
ring automorphisms; subfields;
extending applicability (even to
some curves!) via group maps.

Which ECC *fields* do we use?

2005 Bernstein: prime fields "have the virtue of minimizing the number of security concerns for elliptic-curve cryptography."

2005 ECRYPT key-sizes report: "Some general concerns exist about possible future attacks . . . As a first choice, we recommend curves over prime fields." No extra automorphisms.

Which ECC *fields* do we use?

2005 Bernstein: prime fields "have the virtue of minimizing the number of security concerns for elliptic-curve cryptography."

2005 ECRYPT key-sizes report: "Some general concerns exist about possible future attacks . . . As a first choice, we recommend curves over prime fields." No extra automorphisms.

Imagine a response: "That's premature! $E(\mathbf{F}_{2^n})$ isn't broken!"

Last example: 2013 Garg–Gentry–Halevi–Raykova–Sahai–Waters "Candidate indistinguishability obfuscation and functional encryption for all circuits".

UCLA press release: "According to Sahai, previously developed techniques for obfuscation presented only a 'speed bump,' forcing an attacker to spend some effort, perhaps a few days, trying to reverse-engineer the software. The new system, he said, puts up an 'iron wall' . . . a game-change in the field of cryptography."

2013 Bernstein: "The flagship cryptographic conferences are full of this sort of shit, and, if this is the best defense that the world has against the U.S. National Security Agency, we're screwed."

2013 Bernstein: "The flagship cryptographic conferences are full of this sort of shit, and, if this is the best defense that the world has against the U.S. National Security Agency, we're screwed."

2016 Miles–Sahai–Zhandry: "We exhibit two simple programs that are functionally equivalent, and show how to efficiently distinguish between the obfuscations of these two programs."

So Sahai's claimed "iron wall" is just another "speed bump".

Classic NTRU

Standardize prime p; e.g. 743.

Also standardize q; e.g. 2048.

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

Receiver chooses small $f, g \in \mathcal{R}$.

(Some invertibility requirements.)

Public key $h = 3g/f \bmod q$.

Sender chooses small $m, r \in \mathcal{R}$.

Ciphertext $c = m + hr \bmod q$.

Classic NTRU

Standardize prime p; e.g. 743.

Also standardize q; e.g. 2048.

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

Receiver chooses small $f, g \in \mathcal{R}$.

(Some invertibility requirements.)

Public key $h = 3g/f \bmod q$.

Sender chooses small $m, r \in \mathcal{R}$.

Ciphertext $c = m + hr \bmod q$.

Multiply by $f \bmod q$: $\quad fc \bmod q$.

Classic NTRU

Standardize prime p; e.g. 743.

Also standardize q; e.g. 2048.

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

Receiver chooses small $f, g \in \mathcal{R}$.

(Some invertibility requirements.)

Public key $h = 3g/f \bmod q$.

Sender chooses small $m, r \in \mathcal{R}$.

Ciphertext $c = m + hr \bmod q$.

Multiply by $f \bmod q$: $fc \bmod q$.

Use smallness: $fm + 3gr$.

Classic NTRU

Standardize prime p; e.g. 743.

Also standardize q; e.g. 2048.

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

Receiver chooses small $f, g \in \mathcal{R}$.

(Some invertibility requirements.)

Public key $h = 3g/f \bmod q$.

Sender chooses small $m, r \in \mathcal{R}$.

Ciphertext $c = m + hr \bmod q$.

Multiply by $f \bmod q$: $fc \bmod q$.

Use smallness: $fm + 3gr$.

Reduce mod 3: $fm \bmod 3$.

Classic NTRU

Standardize prime p; e.g. 743.

Also standardize q; e.g. 2048.

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

Receiver chooses small $f, g \in \mathcal{R}$.

(Some invertibility requirements.)

Public key $h = 3g/f \bmod q$.

Sender chooses small $m, r \in \mathcal{R}$.

Ciphertext $c = m + hr \bmod q$.

Multiply by $f \bmod q$: $fc \bmod q$.

Use smallness: $fm + 3gr$.

Reduce mod 3: $fm \bmod 3$.

Divide by $f \bmod 3$: m.

1998 Hoffstein–Pipher–Silverman introduced this system.

Many subsequent NTRU papers: meet-in-the-middle attacks, lattice attacks, hybrid attacks; chosen-ciphertext attacks; decryption-failure attacks; complicated padding systems; variations for efficiency; parameter selection.

Also many ideas that in retrospect were small tweaks of NTRU: e.g., homomorphic encryption.

Unnecessary structures in NTRU

Attacker can evaluate
public polynomials h, c at 1.

Compatible with addition and
multiplication mod $x^p - 1$:
$f(1)h(1) = 3g(1)$ in $\mathbf{Z}/q$;
$c(1) = m(1) + h(1)r(1)$ in $\mathbf{Z}/q$.

Unnecessary structures in NTRU

Attacker can evaluate
public polynomials h, c at 1.

Compatible with addition and
multiplication mod $x^p - 1$:
$f(1)h(1) = 3g(1)$ in $\mathbf{Z}/q$;
$c(1) = m(1) + h(1)r(1)$ in $\mathbf{Z}/q$.

One way to exploit this:
$c(1), h(1)$ are visible; $r(1)$ is
guessable, sometimes standard.
Attacker scans many ciphertexts
to find some with large $m(1)$.
Uses this to speed up m search.

NTRU complicates m selection so that $m(1)$ is never large.

Limits impact of the attack.

NTRU complicates m selection so that $m(1)$ is never large. Limits impact of the attack.

Better: replace NTRU's $\mathbf{Z}[x]/(x^p - 1)$ with $\mathbf{Z}[x]/\Phi_p$. Recall $\Phi_p = (x^p - 1)/(x - 1)$.

NTRU complicates m selection
so that $m(1)$ is never large.
Limits impact of the attack.

Better: replace NTRU's
$\mathbf{Z}[x]/(x^p - 1)$ with $\mathbf{Z}[x]/\Phi_p$.
Recall $\Phi_p = (x^p - 1)/(x - 1)$.

Can view poly m mod $x^p - 1$
as two parts: $m(1)$; m mod Φ_p.
Compatible with add, mult.
Why include $m(1)$ here?
Doesn't seem to help security.

NTRU complicates m selection so that $m(1)$ is never large. Limits impact of the attack.

Better: replace NTRU's $\mathbf{Z}[x]/(x^p - 1)$ with $\mathbf{Z}[x]/\Phi_p$. Recall $\Phi_p = (x^p - 1)/(x - 1)$.

Can view poly m mod $x^p - 1$ as two parts: $m(1)$; m mod Φ_p. Compatible with add, mult. Why include $m(1)$ here? Doesn't seem to help security.

Or use other irreds. Ring-LWE typically uses $\Phi_{2048} = x^{1024} + 1$.

More generally: Attacker applies any ring map $(\mathbf{Z}/q)[x]/P \to T$ to the equations $h = 3g/f$ and $c = m + hr$ in $(\mathbf{Z}/q)[x]/P$.

More generally: Attacker applies any ring map $(\mathbf{Z}/q)[x]/P \to T$ to the equations $h = 3g/f$ and $c = m + hr$ in $(\mathbf{Z}/q)[x]/P$.

e.g. typically $q = 2048$ in NTRU. Have 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. Can attacker exploit these?

Maybe. Complicated. See 2004 Smart–Vercauteren–Silverman.

Ring-LWE religion, version 1: For "provable security", take prime q so that P splits completely in $\mathbf{Z}[x]/q$; i.e., have n different ring maps $(\mathbf{Z}/q)[x]/P \to \mathbf{Z}/q$.

Ring-LWE religion, version 1: For "provable security", take prime q so that P splits completely in $\mathbf{Z}[x]/q$; i.e., have n different ring maps $(\mathbf{Z}/q)[x]/P \to \mathbf{Z}/q$.

Do these maps damage security? Fast attacks in some cases: 2014 Eisenträger–Hallgren–Lauter, 2015 Elias–Lauter–Ozman–Stange, 2016 Chen–Lauter–Stange.

Fast non-q-dependent attack by 2016 Castryck–Iliashenko–Vercauteren breaks 2015 ELOS cases but not 2016 CLS cases.

Ring-LWE religion, version 2 (2012 Langlois–Stehlé): "We prove that the arithmetic form of the modulus q is irrelevant to the computational hardness of LWE and RLWE."

Ring-LWE religion, version 2 (2012 Langlois–Stehlé): "We prove that the arithmetic form of the modulus q is irrelevant to the computational hardness of LWE and RLWE."

Basic idea: "modulus switching" from $\mathbf{Z}/q$ to $\mathbf{Z}/q'$. Attacker multiplies by q'/q and rounds.

Ring-LWE religion, version 2 (2012 Langlois–Stehlé): "We prove that the arithmetic form of the modulus q is irrelevant to the computational hardness of LWE and RLWE."

Basic idea: "modulus switching" from $\mathbf{Z}/q$ to $\mathbf{Z}/q'$. Attacker multiplies by q'/q and rounds.

But rounding adds noise, making attacks harder!
The proof *limits* security gap but does not eliminate it.

We recommend: Take irred P that remains irred in $(\mathbf{Z}/q)[x]$; i.e., choose **inert modulus** q.

Field $(\mathbf{Z}/q)[x]/P$. No ring map to any smaller nonzero ring.

We recommend: Take irred P that remains irred in $(\mathbf{Z}/q)[x]$; i.e., choose **inert modulus** q.

Field $(\mathbf{Z}/q)[x]/P$. No ring map to any smaller nonzero ring.

So far this is compatible with Ring-LWE religion, version 2.

We recommend: Take irred P that remains irred in $(\mathbf{Z}/q)[x]$; i.e., choose **inert modulus** q.

Field $(\mathbf{Z}/q)[x]/P$. No ring map to any smaller nonzero ring.

So far this is compatible with Ring-LWE religion, version 2.

But we also recommend heresy: take P with **prime degree** p and with **large Galois group**, specifically S_p, size $p!$.
Good example: $P = x^p - x - 1$.

2014.02, our 2nd announcement: To eliminate "worrisome" structures, use "a number field of prime degree, so that the only subfield is $\mathbf{Q}$" and "an irreducible polynomial $x^p - x - 1$ with a very large Galois group, so that the number field is very far from having automorphisms".

2014.02, our 2nd announcement:
To eliminate "worrisome"
structures, use "a number field
of prime degree, so that the only
subfield is $\mathbf{Q}$" and "an irreducible
polynomial $x^p - x - 1$ with a
very large Galois group, so that
the number field is very far from
having automorphisms".

Subsequent attacks against
several lattice-based systems
have exploited these structures
and have not been extended
to our recommended rings.

2014.10 Campbell–Groves–Shepherd describe an ideal-lattice-based system “Soliloquy”; claim quantum poly-time key recovery.

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.

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.

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).

SV/Soliloquy parameter:

$k \geq 1$. Define $\mathcal{R} = \mathbf{Z}[x]/\Phi_{2^k}$.

Public key: prime q and $c \in \mathbf{Z}/q$.

Secret key: short element $g \in \mathcal{R}$
with $g\mathcal{R} = q\mathcal{R} + (x - c)\mathcal{R}$;
i.e., short generator
of the ideal $q\mathcal{R} + (x - c)\mathcal{R}$.

SV/Soliloquy parameter:
$k \geq 1$. Define $\mathcal{R} = \mathbf{Z}[x]/\Phi_{2^k}$.

Public key: prime q and $c \in \mathbf{Z}/q$.

Secret key: short element $g \in \mathcal{R}$
with $g\mathcal{R} = q\mathcal{R} + (x - c)\mathcal{R}$;
i.e., short generator
of the ideal $q\mathcal{R} + (x - c)\mathcal{R}$.

But wait, isn't it known how to
compute a generator of an ideal?
See, e.g., 1993 Cohen textbook
"A course in computational
algebraic number theory".

Smart–Vercauteren dismiss this as taking exponential time.

Smart–Vercauteren dismiss this as taking exponential time.

It actually takes subexponential time. Same basic idea as NFS.

Smart–Vercauteren dismiss this as taking exponential time.

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 dismiss this as taking exponential time.

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, building on 2014 Eisenträger–Hallgren–Kitaev–Song: different algorithm that takes quantum poly time.

Smart–Vercauteren also dismiss this generator as not being *short*.

Have ideal I of $\mathcal{R}$.
Want short g with $g\mathcal{R} = I$.
Have g' with $g'\mathcal{R} = I$.
Know $g' = ug$ for some $u \in \mathcal{R}^*$.
But how do we find u?

Smart–Vercauteren also dismiss this generator as not being *short*.

Have ideal I of $\mathcal{R}$.
Want short g with $g\mathcal{R} = I$.
Have g' with $g'\mathcal{R} = I$.
Know $g' = ug$ for some $u \in \mathcal{R}^*$.
But how do we find u?

$\operatorname{Log} g' = \operatorname{Log} u + \operatorname{Log} g$
where Log is Dirichlet's log map.

Dirichlet's unit theorem:
$\operatorname{Log} \mathcal{R}^*$ is a lattice, known dim.

Finding $\operatorname{Log} u$ is a closest-vector problem in this lattice.

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

$x \mapsto x^3$, $x \mapsto x^5$, $x \mapsto x^7$, etc. are automorphisms of $\mathcal{R} = \mathbf{Z}[x]/\Phi_{2^k}$.

Easy to see $(1-x^3)/(1-x) \in \mathcal{R}^*$.

$x \mapsto x^3$, $x \mapsto x^5$, $x \mapsto x^7$, etc. are automorphisms of $\mathcal{R} = \mathbf{Z}[x]/\Phi_{2^k}$.

Easy to see $(1-x^3)/(1-x) \in \mathcal{R}^*$.

"Cyclotomic units" are defined as $\mathcal{R}^* \cap \left\{\pm x^{e_0} \prod_i (1-x^i)^{e_i}\right\}$.
Weber's conjecture: all elements of $\mathcal{R}^*$ are cyclotomic units.

$x \mapsto x^3$, $x \mapsto x^5$, $x \mapsto x^7$, etc. are automorphisms of $\mathcal{R} = \mathbf{Z}[x]/\Phi_{2^k}$.

Easy to see $(1-x^3)/(1-x) \in \mathcal{R}^*$.

"Cyclotomic units" are defined as $\mathcal{R}^* \cap \left\{\pm x^{e_0} \prod_i (1-x^i)^{e_i}\right\}$.
Weber's conjecture: all elements of $\mathcal{R}^*$ 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; missing from bogus CGS analysis.

Attackers can also use automorphisms in more ways.

2016 Albrecht–Bai–Ducas "A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and Graded Encoding Schemes" use norms $g\sigma(g)$, and independently 2016 Cheon–Jeong–Lee ("The main technique of our algorithm is the reduction of a problem on a field to one in a subfield") use traces $g + \sigma(g)$, where σ is an order-2 automorphism.

We recommend changing
the choice of rings in
ideal-lattice-based cryptography.

Requiring prime degree p
minimizes number of subfields.

Requiring Galois group
S_p maximizes difficulty of
automorphism computations: e.g.,
the smallest field containing all
roots of P has degree $p!$.

All available evidence is that
this rescues *some* systems
and never hurts security.

The importance of efficiency

"If you're so worried about structure, why are you tolerating visible polynomial structure? Use LWE, or classic McEliece!"

The importance of efficiency

"If you're so worried about structure, why are you tolerating visible polynomial structure? Use LWE, or classic McEliece!"

Maybe better security, yes—but huge costs in network traffic. Is this affordable?

The importance of efficiency

"If you're so worried about structure, why are you tolerating visible polynomial structure? Use LWE, or classic McEliece!"

Maybe better security, yes—
but huge costs in network traffic.
Is this affordable?

If it is, would we gain more security from larger polynomials?
Larger impact on known attacks, maybe also on unknown attacks.
Not clear what to recommend.

Conventional wisdom:

Rings $(\mathbf{Z}/q)[x]/\Phi_{2^k}$
with $q \bmod 2^{k+1} = 1$ allow
extremely fast FFT-based mults.

NTRU Prime rings will be
several times slower.

Is this affordable? etc.

Conventional wisdom:

Rings $(\mathbf{Z}/q)[x]/\Phi_{2^k}$
with $q \bmod 2^{k+1} = 1$ allow
extremely fast FFT-based mults.
NTRU Prime rings will be
several times slower.
Is this affordable? etc.

But we have shown that
an optimized combination of
Karatsuba and Toom is also
extremely fast at crypto sizes.
Hard to find any applications
that will notice the differences.
And we *improve* network traffic.

What you find in paper

Streamlined NTRU Prime:
an optimized cryptosystem.

The design space of
lattice-based encryption.

Security of Streamlined NTRU Prime: meet-in-the-middle attacks, lattice attacks, etc.

Parameters.

Public-key encryption vs. unauthenticated key exchange.

And more!