Comparing proofs of security for lattice-based encryption

Daniel J. Bernstein

Primary objective of this paper:
Make a **complete plan**
for **thorough security reviews**
of 36 target KEMs.

Much harder: Do the reviews!
Complete plan is framework
to evaluate which pieces are done,
and to coordinate further efforts.
KEMs vary in what's needed.

The target KEMs (all proposed for wide deployment, IND-CCA2):

frodo	640, 976, 1344.
kyber	512, 768, 1024.
lac	128, 192, 256.
newhope	512, 1024.
ntru	hps2048509, hps2048677, hps4096821, hrss701.
ntrulpr	653, 761, 857.
round5n1	1, 3, 5.
round5nd	1.0d, 3.0d, 5.0d, 1.5d, 3.5d, 5.5d.
saber	light, main, fire.
sntrup	653, 761, 857.
threebears	baby, mama, papa.

One categorization of the KEMs:

`frodo`	Product NTRU.
`kyber`	Product NTRU.
`lac`	Product NTRU.
`newhope`	Product NTRU.
`ntru`	Quotient NTRU.
`ntrulpr`	Product NTRU.
`round5n1`	Product NTRU.
`round5nd`	Product NTRU.
`saber`	Product NTRU.
`sntrup`	Quotient NTRU.
`threebears`	Product NTRU.

An oversimplified plan

Plan: Verify the security proofs—make sure there are no mistakes.

An oversimplified plan

Plan: Verify the security proofs—make sure there are no mistakes.

Why verification is important: e.g., Asiacrypt 2004 Rogaway "OCB2" was standardized in 2009, completely broken in 2018. The attack exploited proof error.

An oversimplified plan

Plan: Verify the security proofs— make sure there are no mistakes.

Why verification is important: e.g., Asiacrypt 2004 Rogaway "OCB2" was standardized in 2009, completely broken in 2018. The attack exploited proof error.

I did some sanity checks (*tiny* part of full verification!) and found unproven theorems claimed by `frodo`, `round5n1`, `round5nd`, `saber`; also wrong hypotheses for `newhope` theorem.

Strategy to eliminate proof errors:
explain all of the target proofs
to a thoroughly audited program
that completely verifies proofs.

Strategy to eliminate proof errors: explain all of the target proofs to a thoroughly audited program that completely verifies proofs.

My assessment of this strategy:

- Status today: ≈0% completed.

Strategy to eliminate proof errors: explain all of the target proofs to a thoroughly audited program that completely verifies proofs.

My assessment of this strategy:

- Status today: ≈0% completed.
- Progress is painful and slow. Will we even reach 1% before post-quantum standardization?

Strategy to eliminate proof errors: explain all of the target proofs to a thoroughly audited program that completely verifies proofs.

My assessment of this strategy:

- Status today: ≈0% completed.
- Progress is painful and slow. Will we even reach 1% before post-quantum standardization?
- Easier-to-use proof tools could make strategy work.

Strategy to eliminate proof errors: explain all of the target proofs to a thoroughly audited program that completely verifies proofs.

My assessment of this strategy:

- Status today: ≈0% completed.
- Progress is painful and slow. Will we even reach 1% before post-quantum standardization?
- Easier-to-use proof tools could make strategy work.

Backup strategies: Clean up proofs. Check proofs by hand. Track bug categories, as in code.

Why call this “oversimplified”?

What “security proofs” prove is not actually security.

Why call this “oversimplified”?

What “security proofs” prove
is not actually security.

Even with correct proofs,
there are still risks of attacks.
We all rely on cryptanalysis
for analyzing remaining risks.

Why call this “oversimplified”?

What “security proofs” prove
is not actually security.

Even with correct proofs,
there are still risks of attacks.
We all rely on cryptanalysis
for analyzing remaining risks.

Revised plan:
1. Verify the “security proofs”.
2. Verify the cryptanalysis
of the risks left by the proofs.

Again clean up; check by hand;
track failure categories.

Are attack-cost analyses correct?
How thorough is exploration
of space of optimizations?
How thorough is the study of
claimed barriers to speedups
that work for similar problems?
Do the cryptanalytic targets
match the proof risks? etc.

Long history of failures: e.g.,
NSA overstated DES attack cost;
$L(1/2)$ optimality conjecture
for factorization was wrong;
TLS Triple-DES-CBC was broken
without Triple-DES attack; etc.

Why bother with proofs?

Plan without proofs is simpler:
Verify cryptanalysis of the KEMs.

Why bother with proofs?

Plan without proofs is simpler:
Verify cryptanalysis of the KEMs.

But sometimes the proofs
reduce cost of cryptanalysis.

Why bother with proofs?

Plan without proofs is simpler:
Verify cryptanalysis of the KEMs.

But sometimes the proofs
reduce cost of cryptanalysis.

Sometimes this outweighs
cost to verify proofs: reduces
cost of thorough security review.
Hopefully less chance of disaster.

Why bother with proofs?

Plan without proofs is simpler:
Verify cryptanalysis of the KEMs.

But sometimes the proofs
reduce cost of cryptanalysis.

Sometimes this outweighs
cost to verify proofs: reduces
cost of thorough security review.
Hopefully less chance of disaster.

This paper's verification plan
skips proofs that clearly fail
to reduce cost of cryptanalysis:
e.g., `frodo` seed "reduction".

Risks not ruled out by proofs

A “security proof” guarantees security level λ for system X against all attacks of type T assuming security level λ' for underlying problem P.

Risks not ruled out by proofs

A “security proof” guarantees security level λ for system X against all attacks of type T assuming security level λ' for underlying problem P.

Risk #1: P does not reach security level λ'.

Risks not ruled out by proofs

A "security proof" guarantees security level λ for system X against all attacks of type T assuming security level λ' for underlying problem P.

Risk #1: P does not reach security level λ'.

Risk #2 (looseness): λ is below claimed security level of X.

Risks not ruled out by proofs

A "security proof" guarantees security level λ for system X against all attacks of type T assuming security level λ' for underlying problem P.

Risk #1: P does not reach security level λ'.

Risk #2 (looseness): λ is below claimed security level of X.

Risk #3: There are faster attacks outside type T.

Risks not ruled out by proofs

A “security proof” guarantees security level λ for system X against all attacks of type T assuming security level λ' for underlying problem P.

Risk #1: P does not reach security level λ'.

Risk #2 (looseness): λ is below claimed security level of X.

Risk #3: There are faster attacks outside type T.

Risk #4: Proof is incorrect.

Targets for lattice cryptanalysis

Attack OW-Passive (“OW-CPA”) security of the 36 core PKEs.

Targets for lattice cryptanalysis

Attack OW-Passive ("OW-CPA") security of the 36 core PKEs.

For some targets: Attack IND-CPA security of core PKEs.

Targets for lattice cryptanalysis

Attack OW-Passive ("OW-CPA") security of the 36 core PKEs.

For some targets: Attack IND-CPA security of core PKEs.

For some targets: Attack pseudorandom multipliers.

Targets for lattice cryptanalysis

Attack OW-Passive (“OW-CPA”) security of the 36 core PKEs.

For some targets: Attack IND-CPA security of core PKEs.

For some targets: Attack pseudorandom multipliers.

For some targets: KEM proofs are loose. Find faster attacks.

Targets for lattice cryptanalysis

Attack OW-Passive (“OW-CPA”) security of the 36 core PKEs.

For some targets: Attack IND-CPA security of core PKEs.

For some targets: Attack pseudorandom multipliers.

For some targets: KEM proofs are loose. Find faster attacks. Also, some KEM “proofs” rely on unproven conjectures.

Targets for lattice cryptanalysis

Attack OW-Passive ("OW-CPA") security of the 36 core PKEs.

For some targets: Attack IND-CPA security of core PKEs.

For some targets: Attack pseudorandom multipliers.

For some targets: KEM proofs are loose. Find faster attacks. Also, some KEM "proofs" rely on unproven conjectures.

For all targets: KEM proofs allow non-ROM attacks.

The core PKEs (“P”)

Key generation:

- Table 8.6: Public multiplier G.
- Table 8.7: Short secret a.
- Table 8.8: Public $A \approx aG$.

The core PKEs (“P”)

Key generation:

- Table 8.6: Public multiplier G.
- Table 8.7: Short secret a.
- Table 8.8: Public $A \approx aG$.

Encryption: Short secret b;
public ciphertext $B \approx Gb$
(or $B \approx Gb/3$ or $B \approx 3Gb$).

The core PKEs ("P")

Key generation:

- Table 8.6: Public multiplier G.
- Table 8.7: Short secret a.
- Table 8.8: Public $A \approx aG$.

Encryption: Short secret b;
public ciphertext $B \approx Gb$
(or $B \approx Gb/3$ or $B \approx 3Gb$).

That's it for Quotient NTRU.

The core PKEs (“P”)

Key generation:

- Table 8.6: Public multiplier G.
- Table 8.7: Short secret a.
- Table 8.8: Public $A \approx aG$.

Encryption: Short secret b;
public ciphertext $B \approx Gb$
(or $B \approx Gb/3$ or $B \approx 3Gb$).

That’s it for Quotient NTRU.

More for Product NTRU:

- Table 8.9: Public $C \approx Ab + M$.
- Table 8.10: Secret M.

OW-Passive vs. IND-CPA (“dist”)

Quotient NTRU (`ntru`, `sntrup`) asks for OW-Passive cryptanalysis. 2003 Naor: this is “falsifiable”.

OW-Passive vs. IND-CPA (“dist”)

Quotient NTRU (`ntru`, `sntrup`) asks for OW-Passive cryptanalysis. 2003 Naor: this is “falsifiable”.

Product NTRU (`ntrulpr` and systems not named after NTRU) asks for IND-CPA cryptanalysis. Lower security than OW-Passive? Only “somewhat falsifiable”.

OW-Passive vs. IND-CPA ("dist")

Quotient NTRU (`ntru`, `sntrup`) asks for OW-Passive cryptanalysis. 2003 Naor: this is "falsifiable".

Product NTRU (`ntrulpr` and systems not named after NTRU) asks for IND-CPA cryptanalysis. Lower security than OW-Passive? Only "somewhat falsifiable".

Compare 2006 Goldreich: "What concerns us about" DDH is that "DDH is less simple than DH" making it "harder to evaluate."

Pseudorandom multipliers (“ROM2”)

Product NTRU: convert core PKE into PKE that builds multiplier G pseudorandomly from public seed.

Pseudorandom multipliers ("ROM2")

Product NTRU: convert core PKE into PKE that builds multiplier G pseudorandomly from public seed.

`saber`, `round5n1`, `round5nd` claim that this provably preserves security assuming PRG/PRF.

Pseudorandom multipliers ("ROM2")

Product NTRU: convert core PKE into PKE that builds multiplier G pseudorandomly from public seed.

`saber`, `round5n1`, `round5nd` claim that this provably preserves security assuming PRG/PRF.

I dispute this. Need non-ROM cryptanalysis for all these PKEs. Proofs cover only ROM attacks. Must modify theorem statements.

Pseudorandom multipliers (“ROM2”)

Product NTRU: convert core PKE into PKE that builds multiplier G pseudorandomly from public seed.

`saber`, `round5n1`, `round5nd` claim that this provably preserves security assuming PRG/PRF.

I dispute this. Need non-ROM cryptanalysis for all these PKEs. Proofs cover only ROM attacks. Must modify theorem statements.

`frodo` seed “reduction”: Useless. Still need non-ROM cryptanalysis.

More hashing (“ROM”)

Want the target KEMs
to provide IND-CCA2 security.

The proofs don’t give this,
even assuming security
of the underlying PKEs.
The proofs are limited to
ROM IND-CCA2 attacks.

Issue for Product NTRU
and for Quotient NTRU.

More hashing (“ROM”)

Want the target KEMs to provide IND-CCA2 security.

The proofs don’t give this, even assuming security of the underlying PKEs. The proofs are limited to ROM IND-CCA2 attacks.

Issue for Product NTRU *and* for Quotient NTRU.

For all target KEMs, need non-ROM IND-CCA2 cryptanalysis.

Decryption failures ("fail" / "conj")

2017 Hofheinz–Hövelmanns–Kiltz proofs do not rule out ROM IND-CCA2 attacks with probability $Q\delta$, even if the PKEs are secure.

Q: number of hash calls.
δ: failure probability.

Decryption failures (“fail” / “conj”)

2017 Hofheinz–Hövelmanns–Kiltz proofs do not rule out ROM IND-CCA2 attacks with probability $Q\delta$, even if the PKEs are secure.

Q: number of hash calls.
δ: failure probability.

$\delta = 0$ proven for 10 KEMs: `ntru`, `ntrulpr`, `sntrup`. (Also, simpler ROM IND-CCA2 proof.)

Decryption failures (“fail” / “conj”)

2017 Hofheinz–Hövelmanns–Kiltz proofs do not rule out ROM IND-CCA2 attacks with probability $Q\delta$, even if the PKEs are secure.

Q: number of hash calls.
δ: failure probability.

$\delta = 0$ proven for 10 KEMs: `ntru`, `ntrulpr`, `sntrup`. (Also, simpler ROM IND-CCA2 proof.)

`frodo640`, `kyber512` prove $\delta \le 2^{-128}$ with security goal 2^{128}.

`frodo976` proves $\delta \le 2^{-192}$.

The other 23 KEMs:
Security goal 2^k
without proof that $\delta \le 2^{-k}$.
So need CCA cryptanalysis.

The other 23 KEMs:
Security goal 2^k
without proof that $\delta \le 2^{-k}$.
So need CCA cryptanalysis.

Main issues in these 23 KEMs:

- 14 KEMs do not claim that δ is small enough.

The other 23 KEMs:
Security goal 2^k
without proof that $\delta \le 2^{-k}$.
So need CCA cryptanalysis.

Main issues in these 23 KEMs:

- 14 KEMs do not claim that δ is small enough.

- 15 KEMs conjecture $\delta \le \cdots$ without claiming proof.

The other 23 KEMs:
Security goal 2^k
without proof that $\delta \le 2^{-k}$.
So need CCA cryptanalysis.

Main issues in these 23 KEMs:

- 14 KEMs do not claim that δ is small enough.

- 15 KEMs conjecture $\delta \le \cdots$ without claiming proof.

- 5 KEMs have proofs but do not clearly use correct δ definition. (LEDA uses wrong definition.)

What about quantum attacks?

Consider quantum computers for each cryptanalytic target.

What about quantum attacks?

Consider quantum computers for each cryptanalytic target.

When hashing is involved, analyze three types of attacks:
(1) ROM attacks.
(2) Non-ROM QROM attacks.
(3) Non-QROM attacks.

What about quantum attacks?

Consider quantum computers for each cryptanalytic target.

When hashing is involved, analyze three types of attacks:
(1) ROM attacks.
(2) Non-ROM QROM attacks.
(3) Non-QROM attacks.

Sometimes proofs eliminate #1. Ongoing efforts to extend proofs to similarly eliminate #2. Most QROM proofs are loose, but see 2019 Bindel–Hamburg–Hülsing–Persichetti.

What about multi-user attacks?

Each KEM has quantitative claim of single-user security level λ.

What about multi-user attacks?

Each KEM has quantitative claim of single-user security level λ.

This claim implies quantitative claim λ' of U-user security.
λ' vs. λ: looseness factor U.

What about multi-user attacks?

Each KEM has quantitative claim of single-user security level λ.

This claim implies quantitative claim λ' of U-user security.
λ' vs. λ: looseness factor U.

The only risks of this U-user security claim being broken come from the single-user security claim λ being broken.

What about multi-user attacks?

Each KEM has quantitative claim of single-user security level λ.

This claim implies quantitative claim λ' of U-user security.
λ' vs. λ: looseness factor U.

The only risks of this U-user security claim being broken come from the single-user security claim λ being broken.

As far as I can tell,
none of the target KEMs
claim higher U-user security.