Cold Boot Attacks on Ring & Module-LWE Under the NTT

Martin R. Albrecht, Amit Deo, Kenneth G. Paterson

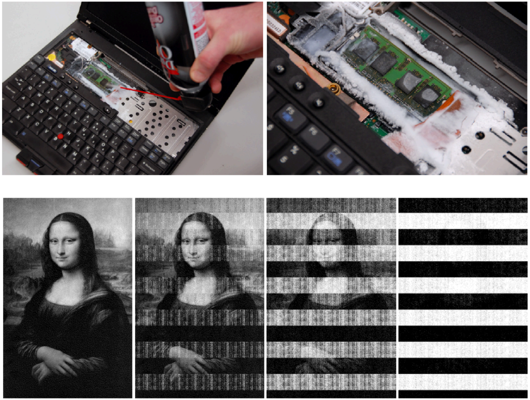
Royal Holloway, University of London

September 12, 2018

Cold boot attack scenario

- Originally investigated by [HSHCPCFAF09]
- An attack method involving physical access to memory storing cryptographic secret keys
- The attacker ejects the memory (lunch-time attack) and plugs into their own machine
- The attacker locates key material in memory and uses data remanence effects [HSHCPCFAF09] to recover the key
- Works on any cryptographic primitive where there is a secret key

Cold boot attacks [HSHCPCFAF09]

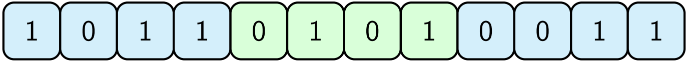


- $< 1\%$ bit flip rate towards ground state after 10 minutes cooling to -50°C
- Limiting case is 0.17% after 1 hour cooling with liquid nitrogen to -196°C

Cold boot attack scenario

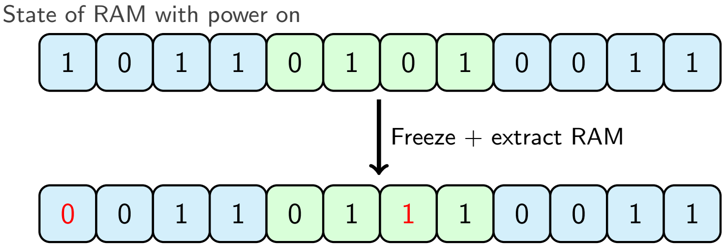
- Bits in RAM decay towards ground state (0/1) on power down
- Cool RAM to extreme temperatures to slow decay

State of RAM with power on



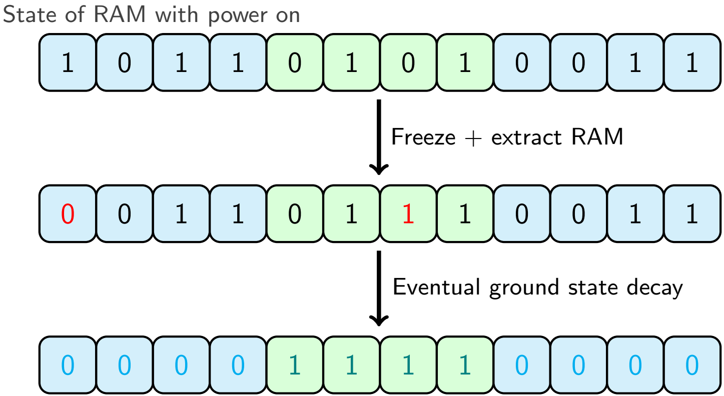
Cold boot attack scenario

- Bits in RAM decay towards ground state (0/1) on power down
- Cool RAM to extreme temperatures to slow decay



Cold boot attack scenario

- Bits in RAM decay towards ground state (0/1) on power down
- Cool RAM to extreme temperatures to slow decay



Cold boot attack flips

- 2 classes of bit flips:
  - Standard bit flips (towards memory ground state) rate ρ_0
  - Retrograde bit flips (away from memory ground state) rate $\rho_1 \approx 0.1\%$
- Assuming half the bits of the key not in ground state

$$\implies \text{\# bit flips} \approx (\text{\# bits in key}) \cdot (\rho_0 + \rho_1)/2$$

- Bit flip rates are written in the form (ρ_0, ρ_1)

Current state-of-the-art

- **DES:** (0.5, 0.001) bit flip rate trivially [HSHCPCFAF09]
- **AES:**
  - AES-128: (0.7,0) bit-flip rate in 1 sec on average [KY10]
  - AES-256: (0.65,0) bit-flip rate in 90 secs on average [Tso09]
- **RSA (1024-bit modulus):**
  (0.4,0.001) bit-flip rate in 2.4 secs on average [PPS12]
- **NTRU:** (0.01,0.001) bit-flip rate in minutes to hours on average for the `ntru-crypto eps449ep1` parameters ($N = 449, \texttt{df} = 134, \texttt{dg} = 149, p = 3, q = 2048$) [PV17]

Post quantum cryptography

- Cryptography resistant to quantum cryptanalytic algorithms
- Plans for wide-spread use and standardisation – NIST process
- 23 lattice-based proposals, the majority of which are LWE based

Post quantum cryptography

- Cryptography resistant to quantum cryptanalytic algorithms
- Plans for wide-spread use and standardisation – NIST process
- 23 lattice-based proposals, the majority of which are LWE based

Are there effective cold boot attacks on some of the LWE-based contenders?

LWE keys

Notation: $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, n a power-of-two
We focus on the two main efficient variations of LWE:

- **Ring-LWE:**
  - $\text{SecKey} = s \in R_q$
- **Module-LWE:**
  - $\text{SecKey} = \mathbf{s} \in R_q^d$

LWE keys

Notation: $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, n a power-of-two
We focus on the two main efficient variations of LWE:

- **Ring-LWE:**
  - SecKey $= s \in R_q$
- **Module-LWE:**
  - SecKey $= \mathbf{s} \in R_q^d$

Trade-off between d and n:

- MLWE Kyber: $n = 256, d = 3$
- RLWE NewHope: $n = 1024, d = 1$

Practical key storage for ring/module-LWE

- The number theoretic transform (`NTT`) is used for efficiency
- Without `NTT`, polynomial multiplication takes $\mathcal{O}(n^2)$ ops
- With `NTT`, polynomial multiplication takes $\mathcal{O}(n \log n)$ ops
- Polynomials in the secret key **s** often stored using an `NTT`

The NTT cold boot problem

"Decode a noisy NTT" **OR** "Recover s from $\tilde{s} = \mathtt{NTT}_n(s) + \Delta \bmod q$"

- Assumption: We have $\kappa \ll n$ bit flips
- Δ's components have a low Hamming weight binary signed digit representation (BSDR)
- A BSDR of 7 is "1, 0, 0, -1" since $7 = 1 * 8 - 1$
- κ bit flips $\implies$ $BSDR(\Delta)$ has Hamming weight κ
- s has small coefficients

The NTT cold boot problem

"Decode a noisy NTT" **OR** "Recover s from $\tilde{s} = \mathtt{NTT}_n(s) + \Delta \bmod q$"

- Assumption: We have $\kappa \ll n$ bit flips
- Δ's components have a low Hamming weight binary <u>signed</u> digit representation (BSDR)
- A BSDR of 7 is "1, 0, 0, -1" since $7 = 1 * 8 - 1$
- κ bit flips $\implies$ $BSDR(\Delta)$ has Hamming weight κ
- s has small coefficients

MLWE Kyber [Sch+17] dimension: $n = 256, d = 3$
RLWE NewHope [Pop+17] dimension: $n = 1024, d = 1$

Attack overview

"Decode a noisy NTT" **OR** "Recover s from $\tilde{s} = \mathtt{NTT}_n(s) + \Delta \bmod q$"

3 main components:

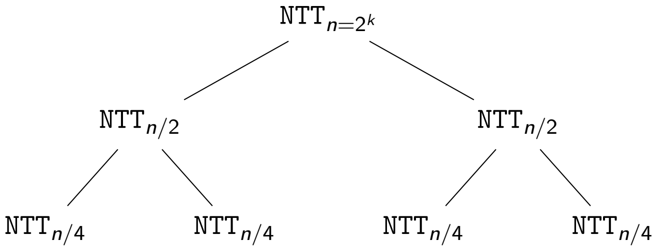
1. Divide and conquer to reduce dimension
2. Work a low-dimensional solution up to solve the problem
3. Lattice + combinatorial attack to solve low dimensional instance

Divide and conquer

Definition

Let ω be a primitive n^{th} root of unity. Then for any $\mathbf{a} \in \mathbb{Z}_q^n$,

$$\mathtt{NTT}(\mathbf{a}) := \sum_{j=0}^{n-1} \omega^{(i+1/2)j} a_j$$



Divide and conquer

For power of two n:

- $\mathbf{a}_e = (a_0, a_2, \ldots, a_{n-2})$
- $\mathbf{a}_o = (a_1, a_3, \ldots, a_{n-1})$

Formulae

For $i = 0, \ldots, n/2 - 1$

$$\mathtt{NTT}_n(\mathbf{a})_i + \mathtt{NTT}_n(\mathbf{a})_{i+n/2} = 2 \cdot \mathtt{NTT}_{n/2}(\mathbf{a}_e)_i$$

$$\mathtt{NTT}_n(\mathbf{a})_i - \mathtt{NTT}_n(\mathbf{a})_{i+n/2} = 2\omega^{i+1/2} \cdot \mathtt{NTT}_{n/2}(\mathbf{a}_o)_i$$

Divide and conquer

Original n-dimensional instance: $\tilde{s} = \mathtt{NTT}_n(s) + \Delta \bmod q$

Folded $n/2$-dimensional instance: For $i = 0, \ldots, n/2 - 1$

$$\tilde{s}_i + \tilde{s}_{i+n/2} = 2 \cdot \mathtt{NTT}_{n/2}(s_e)_i + \overbrace{\left(\Delta_i + \Delta_{i+n/2}\right)}^{(\Delta_+)_i} \quad (1)$$

$$\tilde{s}_i - \tilde{s}_{i+n/2} = 2\omega^{i+1/2} \cdot \mathtt{NTT}_{n/2}(s_o)_i + \underbrace{\left(\Delta_i - \Delta_{i+n/2}\right)}_{(\Delta_-)_i} \quad (2)$$

(1) – the positive fold, (2) – the negative fold

And repeat on the **positive** folded instance ...

Can we reach trivial dimension?

Writing $\Delta = (\Delta_\ell, \Delta_r)$, the error terms after folding once are

- $\Delta_+ = \Delta_\ell + \Delta_r \in \mathbb{Z}_q^{n/2}$
- $\Delta_- = \Delta_\ell - \Delta_r \in \mathbb{Z}_q^{n/2}$

Example

$$\Delta = \dots || \overset{(\Delta_\ell)_i}{1, 0, 0, 0, 0} || \dots || \dots || \overset{(\Delta_r)_i}{0, 0, 0, 0, -1} || \dots$$

$$(\Delta_+)_i = \begin{array}{rr} & 1, 0, 0, 0, \ \ 0 \\ + & 0, 0, 0, 0, -1 \\ \hline & 1, 0, 0, 0, -1 \\ \hline \end{array} \qquad (\Delta_-)_i = \begin{array}{rr} & 1, 0, 0, 0, \ \ 0 \\ - & 0, 0, 0, 0, -1 \\ \hline & -1, 0, 0, 0, \ \ 1 \\ \hline \end{array}$$

Can we reach trivial dimension?

Writing $\Delta = (\Delta_\ell, \Delta_r)$, the error terms after folding once are

- $\Delta_+ = \Delta_\ell + \Delta_r \in \mathbb{Z}_q^{n/2}$
- $\Delta_- = \Delta_\ell - \Delta_r \in \mathbb{Z}_q^{n/2}$

Example

$$\Delta = \dots || \overset{(\Delta_\ell)_i}{1,0,0,0,0} || \dots || \dots || \overset{(\Delta_r)_i}{0,0,0,0,-1} || \dots$$

$$(\Delta_+)_i = \begin{array}{r} 1,0,0,0,\ \ 0 \\ +\ 0,0,0,0,-1 \\ \hline 1,0,0,0,-1 \\ \hline \end{array} \qquad (\Delta_-)_i = \begin{array}{r} 1,0,0,0,\ \ 0 \\ -\ 0,0,0,0,-1 \\ \hline -1,0,0,0,\ \ 1 \\ \hline \end{array}$$

Notes:

- These are less sparse when written in BSDR
- Repeated folding $\rightarrow$ "Δ" term approaches a uniform distribution
- "s" terms stay the same size

Summary of divide and conquer component

top level $\longrightarrow$ $(n = 2^k, \Delta)$

Legend: (dim, Δ)

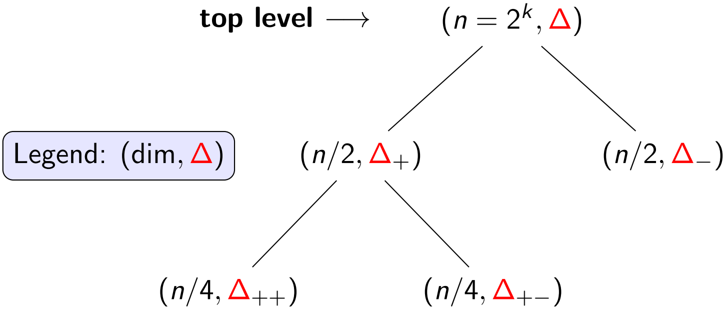
Summary of divide and conquer component

top level $\longrightarrow$ $(n = 2^k, \Delta)$

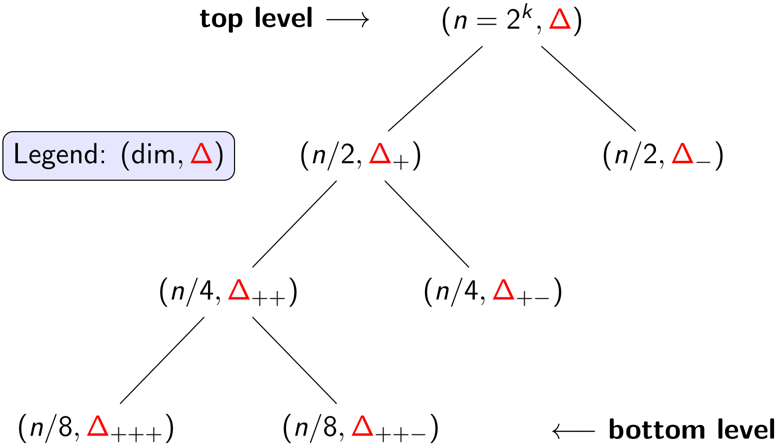
$(n/2, \Delta_+)$ $(n/2, \Delta_-)$

Legend: (dim, Δ)

Summary of divide and conquer component



Summary of divide and conquer component



Working a solution up a level

Instance in $\Delta = (\Delta_\ell, \Delta_r)$ divides into two instances in

- $\Delta_+ = \Delta_\ell + \Delta_r \in \mathbb{Z}_q^{n/2}$
- $\Delta_- = \Delta_\ell - \Delta_r \in \mathbb{Z}_q^{n/2}$

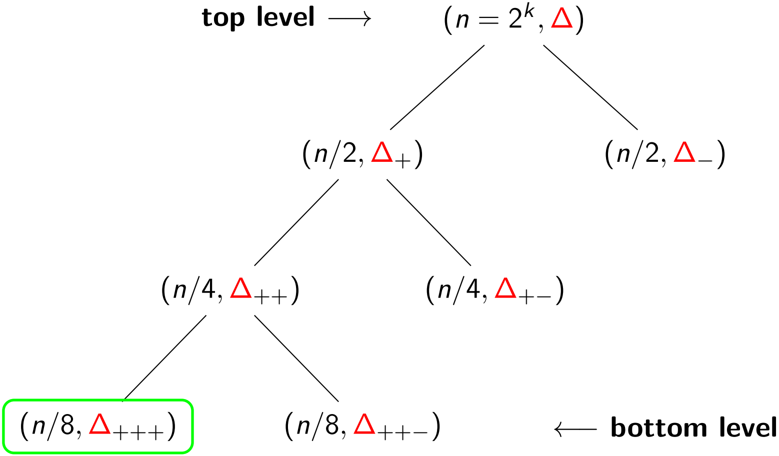
Given Δ_+, guess which bits come from Δ_ℓ and which come from Δ_r to reconstruct Δ. Assuming $\kappa \ll n$, at most 2^κ guesses. [1]

Each guess is verified by plugging the solution into sibling instance.

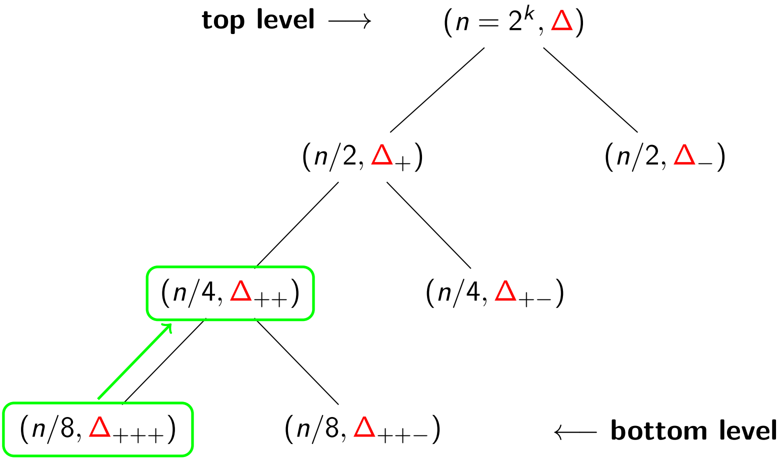
Small complication when bit flips in Δ_ℓ and Δ_r collide!

[1] Compare to $\binom{n \log(q)}{\kappa} \gg 2^\kappa$ guesses for cold boot exhaustive search

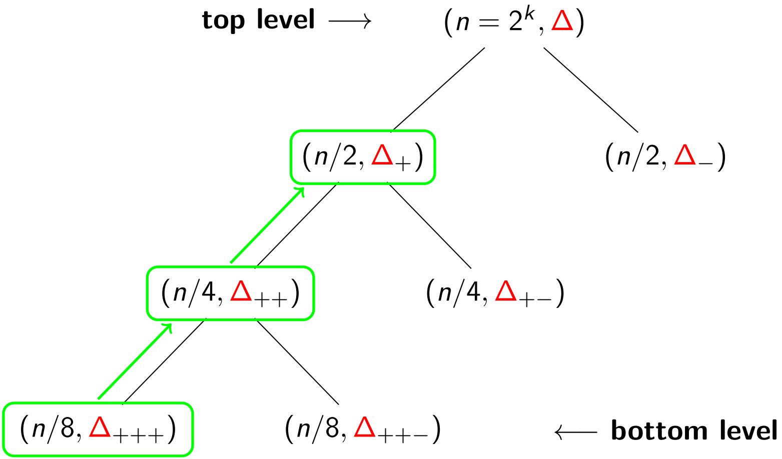
What we have so far



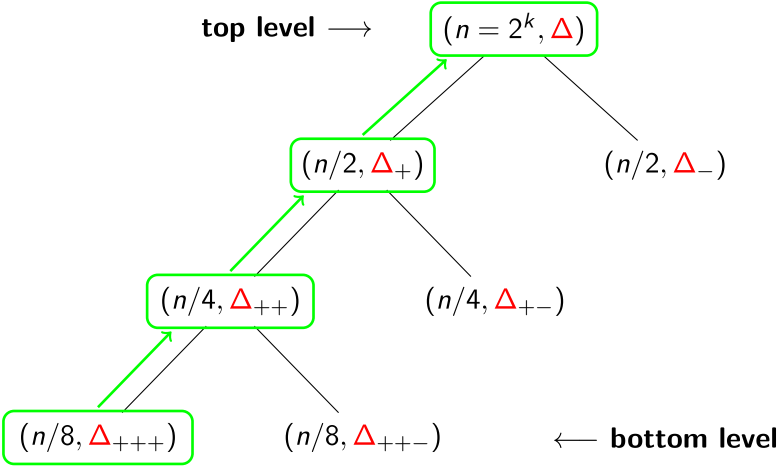
What we have so far



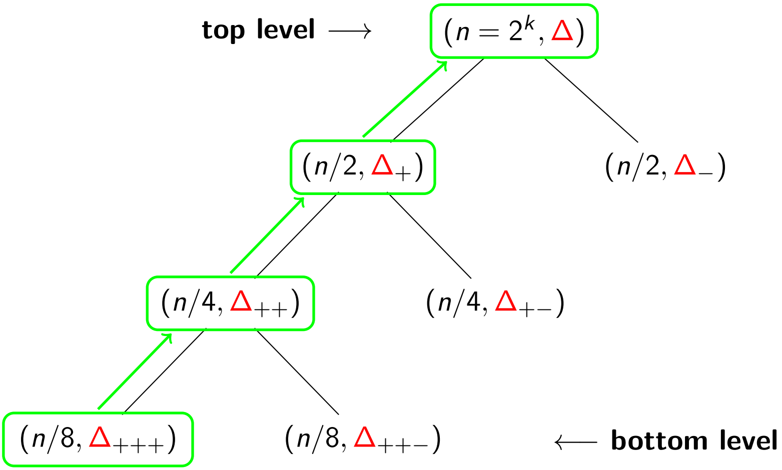
What we have so far



What we have so far



What we have so far



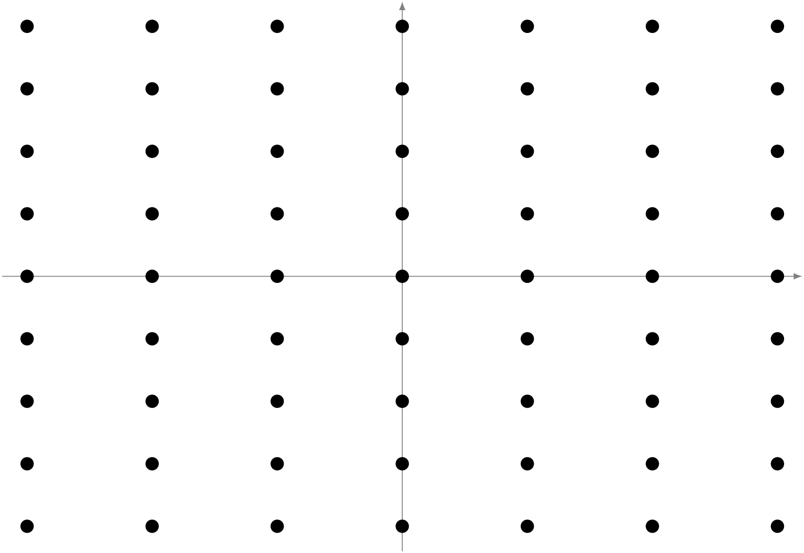
How do we solve the bottom level instance?

Our bottom level instance vs. LWE instances

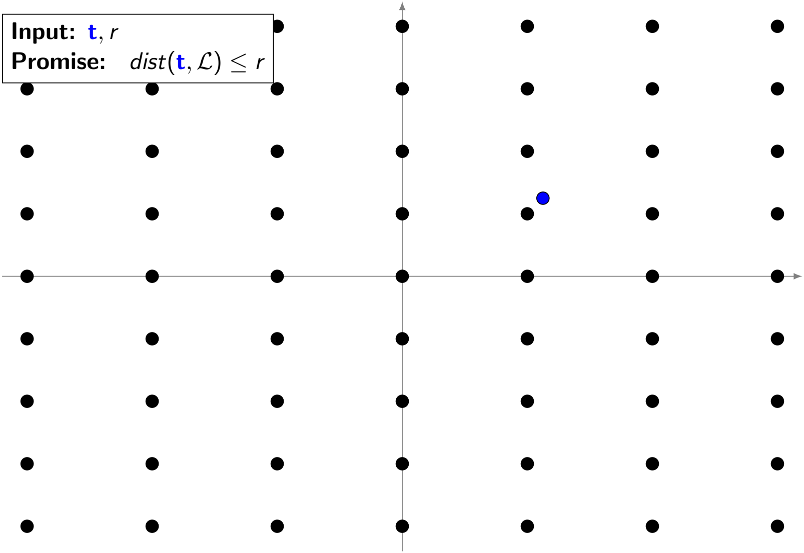
Ours: $\hat{\hat{s}} = \mathtt{NTT}_{n'}^{-1}\Delta + s$	**LWE:** $\mathbf{b} = \mathbf{A}_n s + e$
n' fairly small $(= 32)$	n fairly large $(= 768)$
$\mathtt{NTT}^{-1}$ not random	$\mathbf{A}$ uniform random
s small in ℓ_2	e is small in ℓ_2
Δ not small in ℓ_2	s small in ℓ_2

Despite the differences, let's try to embed our instance into a Bounded Distance Decoding instance

Lattice Background: Bounded Distance Decoding (BDD)

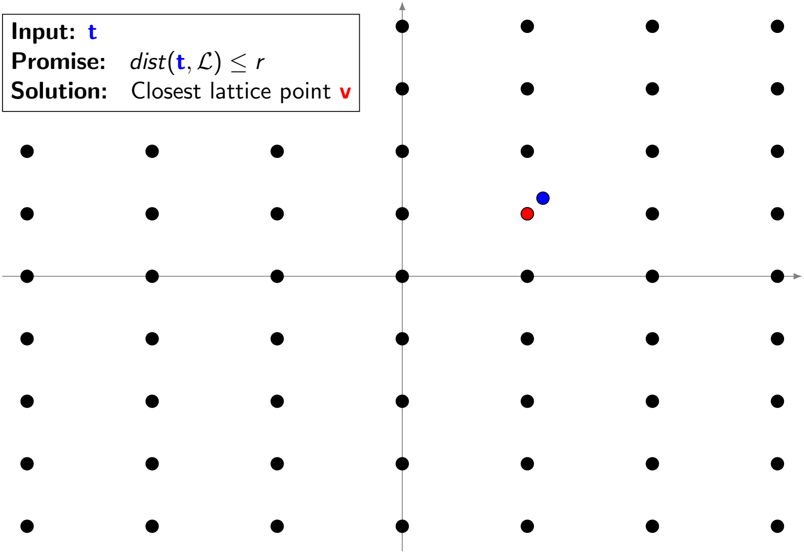


Lattice Background: Bounded Distance Decoding (BDD)



Lattice Background: Bounded Distance Decoding (BDD)

Input: $\mathbf{t}$
Promise: $dist(\mathbf{t}, \mathcal{L}) \leq r$
Solution: Closest lattice point $\mathbf{v}$



Embedding our problem into BDD

Copy the LWE method of:

1. Define target vector $\mathbf{t} := (\mathbf{0}, \hat{\hat{s}}) \in \mathbb{Z}_q^{n'+n'}$
2. Construct lattice
   $\Lambda := \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}_q^{n'+n'} : \mathtt{NTT}^{-1}(\mathbf{x}) + \mathbf{y} = 0 \bmod q\}$
3. Use BDD to find the closest vector in Λ, and hope that the offset vector is $(\Delta, s) \in \mathbb{Z}_q^{n'+n'}$

Embedding our problem into BDD

Copy the LWE method of:

1. Define target vector $\mathbf{t} := (\mathbf{0}, \hat{\hat{s}}) \in \mathbb{Z}_q^{n'+n'}$
2. Construct lattice
   $\Lambda := \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}_q^{n'+n'} : \mathtt{NTT}^{-1}(\mathbf{x}) + \mathbf{y} = 0 \bmod q\}$
3. Use BDD to find the closest vector in Λ, and hope that the offset vector is $(\Delta, s) \in \mathbb{Z}_q^{n'+n'}$

Why/When should we expect to win given a perfect BDD solver?

- Why? $(\Delta, -\mathtt{NTT}^{-1}(\Delta)) \in \Lambda$ and
  $\mathbf{t} - (\Delta, -\mathtt{NTT}^{-1}(\Delta)) = (\Delta, s)$
- When? Expect to win if $||(\Delta, s)||$ is less than half the length of the shortest vector in Λ

Ensuring a successful embedding

"Expect to win if the "offset" $||(\Delta, s)||$ is less than half the length of the shortest vector in Λ"

Ensuring a successful embedding

"Expect to win if the "offset" $||(\Delta, s)||$ is less than half the length of the shortest vector in Λ"

Problem: (Δ, s) is not short!

First step: Consider $2^\ell SDR(\Delta)$ instead of Δ as offset

Fix $\ell := \lceil \log_2(\sqrt{q}) \rceil$ and consider $2^\ell SDR(\Delta)$:

- New lattice is

$$\Lambda' = \{(\mathbf{x}', \mathbf{y}) \in \mathbb{Z}_q^{2n'+n'} : \left(\mathtt{NTT}^{-1} \otimes (1, 2^\ell)\right)(\mathbf{x}') + \mathbf{y} = 0 \bmod q\}$$

- New target vector is $(\mathbf{0}, \hat{\hat{s}}) \in \mathbb{Z}_q^{2n'+n'}$
- The "offset" vector is now $(2^\ell SDR(\Delta), s)$

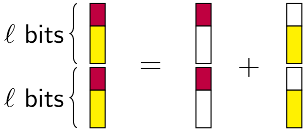
Note:

- Dimension increase is from $2n'$ to $3n'$
- The tensor product introduces terms of the form $(2^\ell, -1, 0, \ldots, 0)$ with length $\approx \sqrt{q}$

Shortening $(2^\ell SDR(\Delta), s)$ offset further

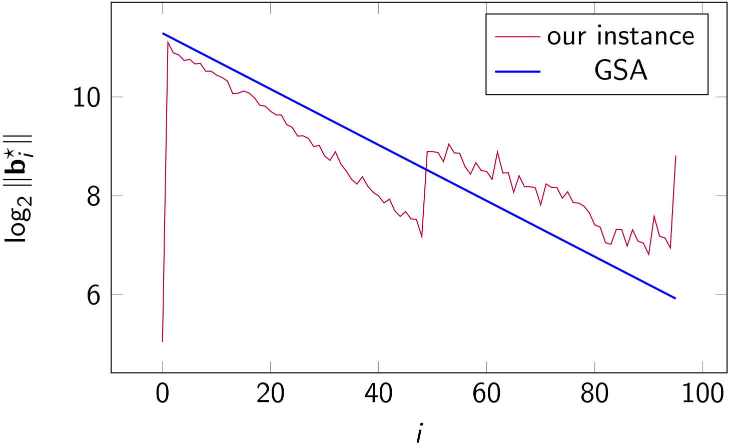
$\ell := \lceil \log_2(\sqrt{q}) \rceil \implies$ each entry of Δ in minimal $2^\ell SDR$ consists of two integers in $\{-2^\ell + 1, \ldots, 0, 2^\ell - 1\}$. Decompose as

$$\Delta_i = \Delta_i^{(\uparrow)} + \Delta_i^{(\downarrow)}.$$



1. Guess bits that contribute the most to length of $2^\ell SDR(\Delta)$.
2. Update the target for our BDD to get new offset $(2^\ell SDR(\Delta^{(\downarrow)}), s)$

Solving BDD in our NTT lattices



- Blue line is expected behaviour of random lattices
- Purple is observed for our lattices

$\therefore$ cannot rely on standard analysis for performance of BDD solver.
Instead we rely on experimental evidence using BDD enumeration.

Overall complexity

Divide and Conquer

Lattice Basis Reduction

BDD Enumeration

Working solution up tree

Overall complexity

~~Divide and Conquer~~	Trivial
~~Lattice Basis Reduction~~	Done once and for all
BDD Enumeration	**Dominates**
~~Working solution up tree~~	2^{κ}

Experimental results[2] using FPLLL[3]

	bit-flip rates		NTT		non-NTT
Scheme	ρ_0	ρ_1	cost	rate	cost
Kyber	0.2%	0.1%	$3 \cdot 2^{21.1}$	95%	$2^{38.7}$
Kyber	1.0%	0.1%	$3 \cdot 2^{43.3}$	91%	$2^{70.3}$
Kyber	1.7%	0.1%	$3 \cdot 2^{62.8}$	89%	$2^{100.1}$
NewHope	0.17%	0.1%	$2^{48.7}$	84%	$2^{53.7}$
NewHope	0.25%	0.1%	$2^{60.6}$	81%	$2^{60.0}$
NewHope	0.32%	0.1%	$2^{70.2}$	81%	$2^{66.1}$

[2] Code available in paper

[3] https://github.com/fplll/fplll

Conclusions

- Structure of the NTT can be exploited by cold boot attackers
- For Kyber parameters, attack complexity of correcting 1% flip rate decreases from 2^{70} to 2^{43} when NTT is used
- For NewHope, not much difference in attack complexity for NTT vs. non-NTT case
- Recommendation: If cold boot attacks are a concern, it is worth not storing secrets using NTT
- Future directions: Solving general LWE like instances with low Hamming weight BSDR secrets, exploiting the rich algebraic structure of NTT's further

References I

- Halderman, J Alex, Seth D Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A Calandrino, Ariel J Feldman, Jacob Appelbaum, and Edward W Felten. "Lest we remember: cold-boot attacks on encryption keys". In: *Communications of the ACM* 52.5 (2009), pp. 91–98.
- Kamal, Abdel Alim and Amr M Youssef. "Applications of SAT solvers to AES key recovery from decayed key schedule images". In: *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*. IEEE. 2010, pp. 216–220.
- Paterson, Kenneth G, Antigoni Polychroniadou, and Dale L Sibborn. "A coding-theoretic approach to recovering noisy RSA keys". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2012, pp. 386–403.
- Paterson, Kenneth G and Ricardo Villanueva-Polanco. "Cold Boot Attacks on NTRU". In: *International Conference in Cryptology in India*. Springer. 2017, pp. 107–125.

References II

Poppelmann, Thomas, Erdem Alkim, Roberto Avanzi, Joppe Bos, Leo Ducas, Antonio de la Piedra, Peter Schwabe, and Douglas Stebila. *NewHope*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. National Institute of Standards and Technology, 2017.

Schwabe, Peter, Roberto Avanzi, Joppe Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehle. *CRYSTALS-KYBER*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. National Institute of Standards and Technology, 2017.

Tsow, Alex. "An improved recovery algorithm for decayed AES key schedule images". In: *International Workshop on Selected Areas in Cryptography*. Springer. 2009, pp. 215–230.