Introduction to post-quantum cryptography and learning with errors

Douglas Stebila
McMaster University
University of Waterloo

Funding acknowledgements:
Summary

• Intro to post-quantum cryptography
• Learning with errors problems
  • LWE, Ring-LWE, Module-LWE, Learning with Rounding, NTRU
  • Search, decision
  • With uniform secrets, with short secrets
• Public key encryption from LWE
  • Regev
  • Lindner–Peikert
• Security of LWE
  • Lattice problems – GapSVP
• KEMs and key agreement from LWE
• Other applications of LWE
• PQ security models
• Transitioning to PQ crypto
Authenticated key exchange + symmetric encryption

Key established using Diffie–Hellman key exchange

Authenticated using RSA digital signatures

Secure channel e.g. TLS

msg → AESEncrypt(k, m) → cipher text → Decrypt(k, c) → msg
Cryptographic building blocks

Public-key cryptography
- RSA signatures
- Elliptic curve Diffie–Hellman key exchange
  - difficulty of factoring
  - difficulty of elliptic curve discrete logarithms

Symmetric cryptography
- AES encryption
- HMAC SHA-256 integrity
  - Cannot be much more efficiently solved by a quantum computer*

Can be solved efficiently by a large-scale quantum computer
When will a large-scale quantum computer be built?

When will a large-scale quantum computer be built?

When will a large-scale quantum computer be built?

“I estimate a 1/7 chance of breaking RSA-2048 by 2026 and a 1/2 chance by 2031.”

— Michele Mosca, November 2015
https://eprint.iacr.org/2015/1075
When will a large-scale quantum computer be built?
Post-quantum cryptography in academia

Conference series

• PQCrypto 2006
• PQCrypto 2008
• PQCrypto 2010
• PQCrypto 2011
• PQCrypto 2013
• PQCrypto 2014
• PQCrypto 2016
• PQCrypto 2017
• PQCrypto 2018
Post-quantum cryptography in government

“IAD will initiate a transition to quantum resistant algorithms in the not too distant future.”

– NSA Information Assurance Directorate, Aug. 2015
### NIST Post-quantum Crypto Project timeline


<table>
<thead>
<tr>
<th>Date</th>
<th>Event Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 2016</td>
<td>Formal call for proposals</td>
</tr>
<tr>
<td>November 2017</td>
<td>Deadline for submissions</td>
</tr>
<tr>
<td></td>
<td>69 submissions</td>
</tr>
<tr>
<td></td>
<td>1/3 signatures, 2/3 KEM/PKE</td>
</tr>
<tr>
<td>3–5 years</td>
<td>Analysis phase</td>
</tr>
<tr>
<td>2 years later (2023–2025)</td>
<td>Draft standards ready</td>
</tr>
</tbody>
</table>
NIST Post-quantum Crypto Project
http://www.nist.gov/pqcrypto

"Our intention is to select a couple of options for more immediate standardization, as well as to eliminate some submissions as unsuitable. ... The goal of the process is not primarily to pick a winner, but to document the strengths and weaknesses of the different options, and to analyze the possible tradeoffs among them."

http://csrc.nist.gov/groups/ST/post-quantum-crypto/faq.html#Q7
Timeline

1995

SHA-1 standardized

NIST

2001

NIST

SHA-2 standardized

2005

NIST

SHA-1 weakened

2016

NIST

Start PQ Crypto project

2017

NIST

Submission deadline

Aug. 2017

Browsers stop accepting SHA-1 certificates

2017

First full SHA-1 collision

Jan. 2017

Browsers stop accepting SHA-1 certificates

2023-25

NIST

Standards ready

2026

Mosca – 1/7 chance of breaking RSA-2048

2031

EU commission – universal quantum computer

2035

Mosca – 1/2 chance of breaking RSA-2048

16 years
Post-quantum crypto

Classical crypto with no known exponential quantum speedup

- **Hash- & symmetric-based**
  - Merkle signatures
  - Sphincs
  - Picnic

- **Code-based**
  - McEliece
  - Niederreiter

- **Multivariate**
  - multivariate quadratic

- **Lattice-based**
  - NTRU
  - learning with errors
  - ring-LWE, ...
  - LWrounding

- **Isogenies**
  - supersingular elliptic curve isogenies
Quantum-resistant crypto
Quantum-safe crypto

Classical post-quantum crypto

- Hash- & Symmetric-based
  - Merkle signatures
  - Sphincs
  - Picnic
- Code-based
  - McEliece
  - Niederreiter
- Multivariate
  - multivariate quadratic
- Lattice-based
  - NTRU
  - learning with errors
  - ring-LWE,
  - LWrounding
- Isogenies
  - supersingular elliptic curve isogenies

Quantum crypto

- Quantum key distribution
- Quantum random number generators
- Quantum channels
- Quantum blind computation
Families of post-quantum cryptography

Hash- & symmetric-based
- Can only be used to make signatures, not public key encryption
- Very high confidence in hash-based signatures, but large signatures required for many signature-systems

Code-based
- Long-studied cryptosystems with moderately high confidence for some code families
- Challenges in communication sizes

Multivariate quadratic
- Variety of systems with various levels of confidence and trade-offs

Lattice-based
- High level of academic interest in this field, flexible constructions
- Can achieve reasonable communication sizes
- Developing confidence

Elliptic curve isogenies
- Specialized but promising technique
- Small communication, slower computation
Learning with errors problems
Solving systems of linear equations

Linear system problem: given blue, find red
Solving systems of linear equations

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>11</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \mathbb{Z}_{13}^{7 \times 4} \]

\[
\begin{array}{cccc}
4 & 1 & 11 & 10 \\
5 & 5 & 9 & 5 \\
3 & 9 & 0 & 10 \\
1 & 3 & 3 & 2 \\
12 & 7 & 3 & 4 \\
6 & 5 & 11 & 4 \\
3 & 3 & 5 & 0 \\
\end{array}
\]

\[
\begin{array}{c}
6 \\
9 \\
11 \\
11 \\
\end{array}
\]

\[ \mathbb{Z}_{13}^{4 \times 1} \]

\[
\begin{array}{c}
4 \\
8 \\
1 \\
10 \\
\end{array}
\]

\[
\begin{array}{c}
4 \\
8 \\
1 \\
10 \\
\end{array}
\]

\[
\begin{array}{c}
12 \\
4 \\
9 \\
\end{array}
\]

\[ \mathbb{Z}_{13}^{7 \times 1} \]

Easily solved using Gaussian elimination (Linear Algebra 101)

Linear system problem: given blue, find red
Learning with errors problem

<table>
<thead>
<tr>
<th>random $\mathbb{Z}_{13}^{7 \times 4}$</th>
<th>secret $\mathbb{Z}_{13}^{4 \times 1}$</th>
<th>small noise $\mathbb{Z}_{13}^{7 \times 1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 1 11 10</td>
<td>6 9 11</td>
<td>0 -1 1</td>
</tr>
<tr>
<td>5 5 9 5</td>
<td></td>
<td>4 7 2</td>
</tr>
<tr>
<td>3 9 0 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 3 3 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 7 3 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 5 11 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 3 5 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\times$ + =

| 2 11 5 12 8 |
Learning with errors problem

\[
\begin{array}{c}
\text{random} \\
\mathbb{Z}_{13}^{7\times 4}
\end{array}
\begin{array}{c}
\times
\end{array}
\begin{array}{c}
\text{secret} \\
\mathbb{Z}_{13}^{4\times 1}
\end{array}
\begin{array}{c}
+ \quad
\text{small noise} \\
\mathbb{Z}_{13}^{7\times 1}
\end{array}
\begin{array}{c}
= \\
\mathbb{Z}_{13}^{7\times 1}
\end{array}
\]

\[
\begin{array}{c}
4 \\
5 \\
3 \\
1 \\
12 \\
6 \\
3
\end{array}
\begin{array}{c}
1 \\
5 \\
9 \\
0 \\
3 \\
5 \\
3
\end{array}
\begin{array}{c}
11 \\
9 \\
0 \\
10 \\
7 \\
11 \\
5
\end{array}
\begin{array}{c}
10 \\
5 \\
10 \\
2 \\
4 \\
5 \\
0
\end{array}
\begin{array}{c}
4 \\
7 \\
2 \\
11 \\
5 \\
12 \\
8
\end{array}
\]

Search LWE problem: given \text{blue}, find \text{red}
Search LWE problem

Let $n$, $m$, and $q$ be positive integers. Let $\chi_s$ and $\chi_e$ be distributions over $\mathbb{Z}$. Let $s \xleftarrow{\$} \chi^n_s$. Let $a_i \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n)$, $e_i \xleftarrow{\$} \chi_e$, and set $b_i \leftarrow \langle a_i, s \rangle + e_i \mod q$, for $i = 1, \ldots, m$.

The search LWE problem for $(n, m, q, \chi_s, \chi_e)$ is to find $s$ given $(a_i, b_i)_{i=1}^m$.

In particular, for algorithm $\mathcal{A}$, define the advantage

$$\text{Adv}_{n, m, q, \chi_s, \chi_e}^{\text{LWE}}(\mathcal{A}) = \Pr \left[ s \xleftarrow{\$} \chi^n_s; a_i \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n); e_i \xleftarrow{\$} \chi_e; \right.$$

$$b_i \leftarrow \langle a_i, s \rangle + e \mod q : \mathcal{A}((a_i, b_i)_{i=1}^m) = s \right].$$

[Regev STOC 2005]
**Decision learning with errors problem**

Decision LWE problem: given blue, distinguish green from random.
Decision LWE problem

Let \( n \) and \( q \) be positive integers. Let \( \chi_s \) and \( \chi_e \) be distributions over \( \mathbb{Z} \). Let \( s \xleftarrow{\$} \chi_s^n \). Define the following two oracles:

- \( O_{\chi_e,s} : a \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n), e \xleftarrow{\$} \chi_e; \text{ return } (a, \langle a, s \rangle + e \mod q) \).
- \( U : a \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n), u \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q); \text{ return } (a, u) \).

The decision LWE problem for \((n, q, \chi_s, \chi_e)\) is to distinguish \( O_{\chi,s} \) from \( U \).

In particular, for algorithm \( \mathcal{A} \), define the advantage

\[
\text{Adv}_{n,q,\chi_s,\chi_e}^{\text{dlwe}}(\mathcal{A}) = \left| \Pr(s \xleftarrow{\$} \mathbb{Z}_q^n : \mathcal{A}^{O_{\chi_e,s}}() = 1) - \Pr(\mathcal{A}^U() = 1) \right|.
\]
Search-decision equivalence

- **Easy fact:** If the search LWE problem is easy, then the decision LWE problem is easy.

- **Fact:** If the decision LWE problem is easy, then the search LWE problem is easy.
  - Requires $nq$ calls to decision oracle
  - Intuition: test the each value for the first component of the secret, then move on to the next one, and so on.
Choice of error distribution

• Usually a discrete Gaussian distribution of width \( s = \alpha q \) for error rate \( \alpha < 1 \)

• Define the Gaussian function

\[
\rho_s(x) = \exp(-\pi \|x\|^2 / s^2)
\]

• The continuous Gaussian distribution has probability density function

\[
f(x) = \rho_s(x) / \int_{\mathbb{R}^n} \rho_s(z) \, dz = \rho_s(x) / s^n
\]
Short secrets

• The secret distribution $\chi_s$ was originally taken to be the uniform distribution

• **Short secrets**: use $\chi_s = \chi_e$

• There's a tight reduction showing that LWE with short secrets is hard if LWE with uniform secrets is hard.

[Applebaum et al., CRYPTO 2009]
Toy example versus real-world example

\[
\mathbb{Z}_{13}^{7\times4}
\]

\[
\begin{array}{cccc}
4 & 1 & 11 & 10 \\
5 & 5 & 9 & 5 \\
3 & 9 & 0 & 10 \\
1 & 3 & 3 & 2 \\
12 & 7 & 3 & 4 \\
6 & 5 & 11 & 4 \\
3 & 3 & 5 & 0 \\
\end{array}
\]

\[
\mathbb{Z}_{215}^{640\times8}
\]

\[
\begin{array}{cccc}
2738 & 3842 & 3345 & 2979 \\
2896 & 595 & 3607 & \\
377 & 1575 & \\
2760 & \\
\ldots & \\
\end{array}
\]

\[640 \times 8 \times 15 \text{ bits} = 9.4 \text{ KiB}\]
Ring learning with errors problem

Each row is the cyclic shift of the row above
Ring learning with errors problem

Each row is the cyclic shift of the row above

... with a special wrapping rule: $x$ wraps to $-x \mod 13$. 

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>12</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Ring learning with errors problem

**random**

$\mathbb{Z}_{13}^{7 \times 4}$

4 1 11 10  

Each row is the cyclic shift of the row above...

...with a special wrapping rule: $x$ wraps to $-x \mod 13$.

So I only need to tell you the first row.
Ring learning with errors problem

\[ \mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle \]

\[
\begin{align*}
4 + 1x + 11x^2 + 10x^3 \\
\times \quad 6 + 9x + 11x^2 + 11x^3 \\
+ \quad + \quad 0 - 1x + 1x^2 + 1x^3 \\
= \quad = \quad 10 + 5x + 10x^2 + 7x^3
\end{align*}
\]
Ring learning with errors problem

\[ \mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle \]

\[
\begin{align*}
4 + 1x + 11x^2 + 10x^3 \\
\times \\
\text{secret} \\
+ \\
\text{small noise} \\
= \\
10 + 5x + 10x^2 + 7x^3
\end{align*}
\]

Search ring-LWE problem: given blue, find red
Search ring-LWE problem

Let $R = \mathbb{Z}[X]/\langle X^n + 1 \rangle$, where $n$ is a power of 2.

Let $q$ be an integer, and define $R_q = R/qR$, i.e., $R_q = \mathbb{Z}_q[X]/\langle X^n + 1 \rangle$.

Let $\chi_s$ and $\chi_e$ be distributions over $R_q$. Let $s \leftarrow \chi_s$. Let $a \leftarrow \mathcal{U}(R_q)$, $e \leftarrow \chi_e$; and set $b \leftarrow as + e$.

The **search ring-LWE problem** for $(n, q, \chi_s, \chi_e)$ is to find $s$ given $(a, b)$.

In particular, for algorithm $A$ define the advantage

$$\text{Adv}_{n,q,\chi_s,\chi_e}^{\text{rlwe}}(A) = \Pr \left[ s \leftarrow \chi_s; a \leftarrow \mathcal{U}(R_q); e \leftarrow \chi_e; b \leftarrow as + e : A(a, b) = s \right] .$$

[Lyubashesky, Peikert, Regev; EUROCRYPT 2010, JACM 2013]
Decision ring-LWE problem

Let $n$ and $q$ be positive integers. Let $\chi_s$ and $\chi_e$ be distributions over $R_q$. Let $s \xleftarrow{\$} \chi_s$. Define the following two oracles:

- $O_{\chi_e,s}$: $a \xleftarrow{\$} \mathcal{U}(R_q), e \xleftarrow{\$} \chi_e$; return $(a, as + e)$.
- $U$: $a, u \xleftarrow{\$} \mathcal{U}(R_q)$; return $(a, u)$.

The decision ring-LWE problem for $(n, q, \chi_s, \chi_e)$ is to distinguish $O_{\chi_e,s}$ from $U$.

In particular, for algorithm $\mathcal{A}$, define the advantage

$$\text{Adv}^{\text{drLWE}}_{n, q, \chi_s, \chi_e}(\mathcal{A}) = \left| \Pr(s \xleftarrow{\$} R_q : \mathcal{A}^{O_{\chi_e,s}}() = 1) - \Pr(\mathcal{A}^U() = 1) \right|.$$
Module learning with errors problem

Every matrix entry is a polynomial in $\mathbb{Z}_q[x]/(x^n + 1)$

Search Module-LWE problem: given blue, find red

Ring-LWE versus Module-LWE

**Ring-LWE**

```
4 1 11 10
3 4 1 11
2 3 4 1
12 2 3 4
9 12 2 3
10 9 12 2
11 10 9 12
```

**Module-LWE**

```
m blocks

Rot(a_1,1) ··· Rot(a_1,d)

Rot(a_m,1) ··· Rot(a_m,d)
```

```
d blocks

n = N/d
```
Learning with rounding problem

<table>
<thead>
<tr>
<th>random $\mathbb{Z}_{13}^{7\times4}$</th>
<th>secret $\mathbb{Z}_{13}^{4\times1}$</th>
<th>$\mathbb{Z}_{13}^{7\times1}$</th>
<th>$\mathbb{Z}_{5}^{7\times1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 1 11 10</td>
<td></td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5 5 9 5</td>
<td></td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>3 9 0 10</td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1 3 3 2</td>
<td></td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>12 7 3 4</td>
<td></td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6 5 11 4</td>
<td></td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>3 3 5 0</td>
<td></td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

$\times$ = $\lfloor \cdot \rfloor_p : \mathbb{Z}_q \rightarrow \mathbb{Z}_p$:
Divide $\mathbb{Z}_q$ into $p$ equal intervals
and map $x$ to the index of its interval

Search LWR problem: given blue, find red

[Banerjee, Peikert, Rosen EUROCRYPT 2012]
LWE versus LWR

**LWE**
- Noise comes from adding an explicit (Gaussian) error term
  \[ \langle a, s \rangle + e \]

**LWR**
- Noise comes from rounding to a smaller interval
  \[ \left\lfloor \langle a, s \rangle \right\rfloor_p \]
- Shown to be as hard as LWE when modulus/error ratio satisfies certain bounds

NTRU problem

For an invertible \( s \in R_q^* \) and a distribution \( \chi \) on \( R \), define \( N_{s,\chi} \) to be the distribution that outputs \( e/s \in R_q \) where \( e \leftarrow \chi \).

The **NTRU learning problem** is: given independent samples \( a_i \in R_q \) where every sample is distributed according to either: (1) \( N_{s,\chi} \) for some randomly chosen \( s \in R_q \) (fixed for all samples), or (2) the uniform distribution, distinguish which is the case.
# Problems

<table>
<thead>
<tr>
<th>Learning with errors</th>
<th>Module-LWE</th>
<th>Search</th>
<th>With uniform secrets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring-LWE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learning with rounding</td>
<td></td>
<td>Decision</td>
<td>With short secrets</td>
</tr>
<tr>
<td>NTRU problem</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Public key encryption from LWE
Public key encryption from LWE

Key generation

\[ A + s + e = b \]

Secret key

Public key

[Lindner, Peikert. CT-RSA 2011]
Public key encryption from LWE

Encryption

\[ A = s' + b \]

\[ e' = b' \]

\[ c = \frac{q}{2} m + v' \]

[Source: Lindner, Peikert. CT-RSA 2011]
Public key encryption from LWE

Decryption

\[ b' = s \]

Ciphertext

\[ v' + \frac{q}{2} m = c \]

Almost the same shared secret mask as the sender used

Secret key

\[ v \]

\[ c - v \approx \frac{q}{2} m \text{ round} \]

\[ m \]
Approximately equal shared secret

The sender uses

\[ v' = s' (A s + e) + e' \]

\[ = s' A s + (s' e + e'') \]

\[ \approx s' A s \]

The receiver uses

\[ v = (s' A + e') s \]

\[ = s' A s + (e' s) \]

\[ \approx s' A s \]
Regev's public key encryption scheme

Let $n, m, q, \chi$ be LWE parameters.

- **KeyGen:** $s \leftarrow \mathbb{Z}_q^n$. $A \leftarrow \mathbb{Z}_q^{m \times n}$. $e \leftarrow \chi(\mathbb{Z}_q)$. $\tilde{b} \leftarrow As + e$. Return $pk \leftarrow (A, b)$, $sk \leftarrow s$.

- **Enc($pk, x \in \{0, 1\}$): $s' \leftarrow \{0, 1\}^m$. $b' \leftarrow s'A$. $v' \leftarrow \langle s', b \rangle$. $c \leftarrow x \cdot \text{encode}(v')$. Return $(b', c)$.

- **Dec($sk, (b', c))$: $v \leftarrow \langle b', s \rangle$. Return $\text{decode}(v)$.
Encode/decode

\[ \text{encode}(x \in \{0, 1\}) \leftarrow x \cdot \left\lceil \frac{q}{2} \right\rceil \]

\[ \text{decode}(\bar{x} \in \mathbb{Z}_q) \leftarrow \begin{cases} 0, & \text{if } \bar{x} \in \left[-\left\lfloor \frac{q}{4} \right\rfloor, \left\lceil \frac{q}{4} \right\rceil \right) \\ 1, & \text{otherwise} \end{cases} \]

[Regev; STOC 2005]
Lindner–Peikert public key encryption

Let $n, q, \chi$ be LWE parameters.

- **KeyGen($\cdot$):** $s \leftarrow \chi(\mathbb{Z}^n)$. $A \leftarrow \mathbb{Z}_{q}^{n \times n}$. $e \leftarrow \chi(\mathbb{Z}^n)$. $\tilde{b} \leftarrow As + e$. Return $pk \leftarrow (A, \tilde{b})$ and $sk \leftarrow s$.

- **Enc($pk, x \in \{0, 1\}$):** $s' \leftarrow \chi(\mathbb{Z}^n)$. $e' \leftarrow \chi(\mathbb{Z}^n)$. $\tilde{b}' \leftarrow s'A + e'$. $e'' \leftarrow \chi(\mathbb{Z})$. $\tilde{v}' \leftarrow \langle s', \tilde{b} \rangle + e''$. $c \leftarrow$ encode$(x) + \tilde{v}'$. Return $ctxt \leftarrow (\tilde{b}', c)$.

- **Dec($sk, (\tilde{b}', c)$):** $v \leftarrow \langle \tilde{b}', s \rangle$. Return decode$(c - v)$.
Correctness

Sender and receiver approximately compute the same shared secret $s' As$

$$\tilde{v}' = \langle s', \tilde{b} \rangle + e'' = s'(As + e) + e'' = s'As + \langle s', e \rangle + e'' \approx s'As$$

$$v = \langle \tilde{b}', s \rangle = (s'A + e')s = s'As + \langle e', s \rangle \approx s'As$$
Difference between Regev and Lindner–Peikert

Regev:

- Bob’s public key is $s'A$ where $s' \leftarrow \{0, 1\}^m$
- Encryption mask is $\langle s', b \rangle$

Lindner–Peikert:

- Bob’s public key is $s'A + e'$ where $s' \leftarrow \chi_e$
- Encryption mask is $\langle s', b \rangle + e''$

In Regev, Bob’s public key is a subset sum instance. In Lindner–Peikert, Bob’s public key and encryption mask is just another LWE instance.
IND-CPA security of Lindner–Peikert
Indistinguishable against chosen plaintext attacks

**Theorem.** If the decision LWE problem is hard, then Lindner–Peikert is IND-CPA-secure. Let $n, q, \chi$ be LWE parameters. Let $\mathcal{A}$ be an algorithm. Then there exist algorithms $\mathcal{B}_1, \mathcal{B}_2$ such that

$$\text{Adv}_{\text{LP}^{\text{ind}-\text{cpa}}[n,q,\chi]}(\mathcal{A}) \leq \text{Adv}_{n,q,\chi}^{\text{dLWE}}(\mathcal{A} \circ \mathcal{B}_1) + \text{Adv}_{n,q,\chi}^{\text{dLWE}}(\mathcal{A} \circ \mathcal{B}_2)$$

[Lindner, Peikert; CT-RSA 2011]
IND-CPA security of Lindner–Peikert

**Game 0:** \( \rightarrow \) Decision-LWE \( \rightarrow \)

1. \( \mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n}) \)
2. \( \mathbf{s}, \mathbf{e} \leftarrow \chi(\mathbb{Z}_q^n) \)
3. \( \tilde{\mathbf{b}} \leftarrow \mathbf{A}s + \mathbf{e} \)
4. \( \mathbf{s}', \mathbf{e}' \leftarrow \chi(\mathbb{Z}_q^n) \)
5. \( \tilde{\mathbf{b}}' \leftarrow \mathbf{s}'\mathbf{A} + \mathbf{e}' \)
6. \( \mathbf{e}'' \leftarrow \chi(\mathbb{Z}_q^n) \)
7. \( \tilde{\nu}' \leftarrow \mathbf{s}'\tilde{\mathbf{b}} + \mathbf{e}'' \)
8. \( c_0 \leftarrow \text{encode}(0) + \tilde{\nu}' \)
9. \( c_1 \leftarrow \text{encode}(1) + \tilde{\nu}' \)
10. \( \mathbf{b}^* \leftarrow \mathcal{U}([0, 1]) \)
11. **return**

\((\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})\)

**Game 1:** \( \rightarrow \) Rewrite \( \rightarrow \)

1. \( \mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n}) \)
2. \( \tilde{\mathbf{b}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \)
3. \( \mathbf{s}' \leftarrow \chi(\mathbb{Z}_q^n) \)
4. \( \tilde{\mathbf{b}}' \leftarrow \mathbf{s}'\mathbf{A} + \mathbf{e}' \)
5. \( \mathbf{e}'' \leftarrow \chi(\mathbb{Z}_q^n) \)
6. \( \tilde{\nu}' \leftarrow \mathbf{s}'\tilde{\mathbf{b}} + \mathbf{e}'' \)
7. \( c_0 \leftarrow \text{encode}(0) + \tilde{\nu}' \)
8. \( c_1 \leftarrow \text{encode}(1) + \tilde{\nu}' \)
9. \( \mathbf{b}^* \leftarrow \mathcal{U}([0, 1]) \)
10. **return**

\((\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})\)

**Game 2:**

1. \( \mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n}) \)
2. \( \tilde{\mathbf{b}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \)
3. \( \mathbf{s}' \leftarrow \chi(\mathbb{Z}_q^n) \)
4. \( \tilde{\mathbf{b}}' \leftarrow \mathbf{s}'\mathbf{A} + \mathbf{e}' \)
5. \( \mathbf{e}'' \leftarrow \chi(\mathbb{Z}_q^{n+1}) \)
6. \( \tilde{\nu}' \leftarrow \mathbf{s}'\tilde{\mathbf{b}} + \mathbf{e}'' \)
7. \( c_0 \leftarrow \text{encode}(0) + \tilde{\nu}' \)
8. \( c_1 \leftarrow \text{encode}(1) + \tilde{\nu}' \)
9. \( \mathbf{b}^* \leftarrow \mathcal{U}([0, 1]) \)
10. **return**

\((\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})\)

[Lindner, Peikert; CT-RSA 2011]
IND-CPA security of Lindner–Peikert

**Game 2:** \[ \rightarrow \text{Decision-LWE} \rightarrow \]

1. \( \mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n}) \)
2. \( \tilde{\mathbf{b}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \)
3. \( s' \leftarrow \chi(\mathbb{Z}_q^n) \)
4. \( \begin{bmatrix} \mathbf{e}' & \mathbf{e}'' \end{bmatrix} \leftarrow \chi(\mathbb{Z}_q^{n+1}) \)
5. \( \begin{bmatrix} \tilde{\mathbf{b}}' \parallel \tilde{\mathbf{v}}' \end{bmatrix} \leftarrow s' [\mathbf{A} \parallel \tilde{\mathbf{b}}] + \begin{bmatrix} \mathbf{e}' \parallel \mathbf{e}'' \end{bmatrix} \)
6. \( c_0 \leftarrow \text{encode}(0) + \tilde{\mathbf{v}}' \)
7. \( c_1 \leftarrow \text{encode}(1) + \tilde{\mathbf{v}}' \)
8. \( b^* \leftarrow \mathcal{U}(\{0, 1\}) \)
9. **return** \( (\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*}) \)

**Game 3:** \[ \rightarrow \text{Rewrite} \rightarrow \]

1. \( \mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n}) \)
2. \( \tilde{\mathbf{b}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \)
3. \( \begin{bmatrix} \tilde{\mathbf{b}}' \parallel \tilde{\mathbf{v}}' \end{bmatrix} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n+1}) \)
4. \( c_0 \leftarrow \text{encode}(0) + \tilde{\mathbf{v}}' \)
5. \( c_1 \leftarrow \text{encode}(1) + \tilde{\mathbf{v}}' \)
6. \( b^* \leftarrow \mathcal{U}(\{0, 1\}) \)
7. **return** \( (\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*}) \)

**Game 4:**

1. \( \mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n}) \)
2. \( \tilde{\mathbf{b}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \)
3. \( \begin{bmatrix} \tilde{\mathbf{b}}' \parallel \tilde{\mathbf{v}}' \end{bmatrix} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n+1}) \)
4. \( b^* \leftarrow \mathcal{U}(\{0, 1\}) \)
5. **return** \( (\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', \tilde{\mathbf{v}}', \tilde{\mathbf{b}}^*) \)

Independent of hidden bit

[Indenner, Peikert; CT-RSA 2011]
Lattice-based KEM/PKEs submitted to NIST

- BabyBear, MamaBear, PapaBear (ILWE)
- CRYSTALS-Kyber (MLWE)
- Ding Key Exchange (RLWE)
- Emblem (LWE, RLWE)
- FrodoKEM (LWE)
- HILA5 (RLWE)
- KCL (MLWE, RLWE)
- KINDI (MLWE)
- LAC (PLWE)
- LIMA (RLWE)
- Lizard (LWE, LWR, RLWE, RLWR)
- Lotus (LWE)
- NewHope (RLWE)
- NTRU Prime (RLWR)
- NTRU HRSS (NTRU)
- NTRUEncrypt (NTRU)
- Round2 (RLWR, LWR)
- Saber (MLWR)
- Titanium (PLWE)

https://estimate-all-the-lwe-ntru-schemes.github.io/docs/
Security of LWE-based cryptography

"Lattice-based"
Hardness of decision LWE – "lattice-based"

- worst-case gap shortest vector problem (GapSVP)
  - poly-time [Regev05, BLPRS13]
- average-case decision LWE
Lattices

Let $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_n\} \subseteq \mathbb{Z}_q^{n \times n}$ be a set of linearly independent basis vectors for $\mathbb{Z}_q^n$. Define the corresponding lattice

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\} .$$

(In other words, a lattice is a set of integer linear combinations.)

Define the minimum distance of a lattice as

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \{0\}} \| \mathbf{v} \| .$$
Lattices

Discrete additive subgroup of $\mathbb{Z}^n$

Equivalently, integer linear combinations of a basis

Lattices

There are many bases for the same lattice – some short and orthogonalish, some long and acute.

Closest vector problem

Given some basis for the lattice and a target point in the space, find the closest lattice point.

Shortest vector problem

Given some basis for the lattice, find the shortest non-zero lattice point.

Shortest vector problem

The **shortest vector problem** (SVP) is: given a basis $B$ for some lattice $\mathcal{L} = \mathcal{L}(B)$, find a shortest non-zero vector, i.e., find $v \in \mathcal{L}$ such that $\|v\| = \lambda_1(\mathcal{L})$.

The **decision approximate shortest vector problem** (GapSVP$_\gamma$) is: given a basis $B$ for some lattice $\mathcal{L} = \mathcal{L}(B)$ where either $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma$, determine which is the case.
Regev's iterative reduction

**Theorem.** [Reg05] For any modulus $q \leq 2^{\text{poly}(n)}$ and any discretized Gaussian error distribution $\chi$ of parameter $\alpha q \geq 2\sqrt{n}$ where $0 < \alpha < 1$, solving the decision LWE problem for $(n, q, \mathcal{U}, \chi)$ with at most $m = \text{poly}(n)$ samples is at least as hard as quantumly solving $\text{GapSVP}_\gamma$ and $\text{SIVP}_\gamma$ on arbitrary $n$-dimensional lattices for some $\gamma = \tilde{O}(n/\alpha)$.

The polynomial-time reduction is extremely non-tight: approximately $O(n^{13})$. 

[Regev; STOC 2005]
Finding short vectors in lattices

LLL basis reduction algorithm

- Finds a basis close to Gram–Schmidt
- Polynomial runtime (in dimension), but basis quality (shortness/orthogonality) is poor

Block Korkine Zolotarev (BKZ) algorithm

- Trade-off between runtime and basis quality
- In practice the best algorithm for cryptographically relevant scenarios
Solving the (approximate) shortest vector problem

The complexity of $\text{GapSVP}_\gamma$ depends heavily on how $\gamma$ and $n$ relate, and get harder for smaller $\gamma$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Approx. factor $\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLL algorithm</td>
<td>$\text{poly}(n)$</td>
<td>$2^{\Omega(n \log \log n / \log n)}$</td>
</tr>
<tr>
<td>various</td>
<td>$2^{\Omega(n \log n)}$</td>
<td>$\text{poly}(n)$</td>
</tr>
<tr>
<td>various</td>
<td>$2^{\Omega(n)}$ time and space</td>
<td>$2^k$</td>
</tr>
<tr>
<td>Sch87</td>
<td>$2^{\Omega(n/k)}$</td>
<td>$\geq \sqrt{n}$</td>
</tr>
<tr>
<td></td>
<td>$\text{NP} \cap \text{co-NP}$</td>
<td>$n^{o(1)}$</td>
</tr>
<tr>
<td></td>
<td>$\text{NP-hard}$</td>
<td></td>
</tr>
</tbody>
</table>

In cryptography, we tend to use $\gamma \approx n$. 
Picking parameters

• Estimate parameters based on runtime of lattice reduction algorithms.

• Based on reductions:
  • Calculate required runtime for GapSVP or SVP based on tightness gaps and constraints in each reduction
  • Pick parameters based on best known GapSVP or SVP solvers or known lower bounds
    • Reductions are typically non-tight (e.g., $n^{13}$); would lead to very large parameters

• Based on cryptanalysis:
  • Ignore tightness in reductions.
  • Pick parameters based on best known LWE solvers relying on lattice solvers.
KEMs and key agreement from LWE
Key encapsulation mechanisms (KEMs)

A key encapsulation mechanism (KEM) consists of three algorithms:

- **KeyGen() \( \rightarrow (pk, sk) \)**: A key generation algorithm that outputs a public key \( pk \) and secret key \( sk \)

- **Encaps(pk) \( \rightarrow (c, k) \)** An encapsulation algorithm that outputs a ciphertext \( c \) and session key \( k \)

- **Decaps(sk, c) \( \rightarrow k \)**: A decapsulation algorithm that outputs a session key \( k \) (or an error symbol)

Security properties for KEMs: IND-CPA, IND-CCA
Key exchange protocols

• A key exchange protocol is an interactive protocol carried out between two parties.
• The goal of the protocol is to output a session key that is indistinguishable from random.

• In authenticated key exchange protocols, the adversary can be active and controls all communications between parties; the parties are assumed to have authentically distributed trusted long-term keys out of band prior to the protocol.
• In unauthenticated key exchange protocols, the adversary can be passive and only obtains transcripts of communications between honest parties.

• IND-CPA KEMs can be viewed as a two flow unauthenticated key exchange protocol.
Basic LWE key agreement (unauthenticated)

Based on Lindner–Peikert LWE public key encryption scheme

public: “big” \( A \) in \( \mathbb{Z}_{q}^{n \times m} \)

Alice
secret:
random “small” \( s, e \) in \( \mathbb{Z}_{q}^{m} \)

Bob
secret:
random “small” \( s', e' \) in \( \mathbb{Z}_{q}^{n} \)

\[ b = As + e \]
\[ b' = s'A + e' \]

shared secret:
\( b's = s'As + e's \approx s'As \)

shared secret:
\( s'b \approx s'As \)

These are only approximately equal \( \Rightarrow \) need rounding
Rounding & reconciliation

- Each coefficient of the polynomial is an integer modulo $q$
- Treat each coefficient independently
- Send a "reconciliation signal" to help with rounding

- Techniques by Ding [Din12] and Peikert [Pei14]
Basic rounding

- Round either to 0 or $q/2$
- Treat $q/2$ as 1

This works most of the time: prob. failure $2^{-10}$.

Not good enough: we need exact key agreement.
Rounding and reconciliation (Peikert)

Bob says which of two regions the value is in: or

If or

If or

[Peikert; PQCrypto 2014]
Rounding and reconciliation (Peikert)

- If $|alice - bob| \leq \frac{q}{8}$, then this always works.

- Security not affected: revealing $\leq \frac{q}{4}$ or $\geq \frac{3q}{4}$ leaks no information.
Exact LWE key agreement (unauthenticated)

Public: “big” $A$ in $\mathbb{Z}_q^{n \times m}$

Alice

- Secret: random “small” $s, e$ in $\mathbb{Z}_q^m$

Bob

- Secret: random “small” $s', e'$ in $\mathbb{Z}_q^n$

$\text{Alice}$

$\text{Bob}$

$b = As + e$

$b' = s'A + e'$, or $\text{round}(b')$

shared secret: $\text{round}(b')$

shared secret: $\text{round}(s'b)$
Exact ring-LWE key agreement (unauthenticated)

public: “big” \( a \) in \( R_q = \mathbb{Z}_q[x]/(x^n+1) \)

Alice

secret:
random “small” \( s, e \) in \( R_q \)

Bob

secret:
random “small” \( s’, e’ \) in \( R_q \)

\[ b = a \cdot s + e \]

\[ b’ = a \cdot s’ + e’, \quad \text{or} \]

shared secret:
\[ \text{round}(s \cdot b’) \]

shared secret:
\[ \text{round}(b \cdot s’) \]
Public key validation

- **No public key validation possible** for basic LWE/ring-LWE public keys

- **Key reuse in LWE/ring-LWE** leads to real attacks following from search-decision equivalence
  - Comment in [Peikert, PQCrypto 2014]
  - Attack described in [Fluhrer, Eprint 2016]

- Need to ensure usage is okay with just passive security (IND-CPA)
- Or construct actively secure (IND-CCA) KEM/PKE/AKE using Fujisaki–Okamoto transform or quantum-resistant variant [Targhi–Unruh, TCC 2016] [Hofheinz et al., Eprint 2017]
An example: FrodoKEM

- **KEM**: Key encapsulation mechanism (simplified key exchange protocol)
- Builds on basic (IND-CPA) LWE public key encryption
- Achieves IND-CCA security against adaptive adversaries
  - By applying a quantum-resistant variant of the Fujisaki–Okamoto transform
- Negligible error rate

- **Simple design:**
  - Free modular arithmetic \( (q = 2^{16}) \)
  - Simple Gaussian sampling
  - Parallelizable matrix-vector operations
  - No reconciliation
  - Simple to code

---

[Bos, Costello, Ducas, Mironov, Naehrig, Nikolaenko, Raghunathan, Stebila. ACM CCS 2016]
[Alkim, Bos, Ducas, Easterbrook, LaMacchia, Longa, Mironov, Naehrig, Nikolaenko, Peikert, Raghunathan, Stebila. FrodoKEM NIST Submission, 2017]
FrodoKEM construction

**Algorithm 9 FrodoPKE.KeyGen.**

**Input:** None.

**Output:** Key pair $(pk, sk) \in (\{0, 1\}^{\text{len}_A} \times \mathbb{Z}_q^{n \times \overline{n}}) \times \mathbb{Z}_q^{n \times \overline{n}}$.

1. Choose a uniformly random seed $\text{seed}_A \leftarrow \mathcal{U}(\{0, 1\}^{\text{len}_A})$.
2. Generate the matrix $A \in \mathbb{Z}_q^{n \times \overline{n}}$ via $A \leftarrow \text{Frodo.Gen}(\text{seed}_A)$.
3. Choose a uniformly random seed $\text{seed}_E \leftarrow \mathcal{U}(\{0, 1\}^{\text{len}_E})$.
4. Sample error matrix $S \leftarrow \text{Frodo.SampleMatrix}(\text{seed}_E, n, \overline{n}, T_X, 1)$.
5. Sample error matrix $E \leftarrow \text{Frodo.SampleMatrix}(\text{seed}_E, n, \overline{n}, T_X, 2)$.
7. return public key $pk \leftarrow (\text{seed}_A, B)$ and secret key $sk \leftarrow S$. 

**FrodoPKE.KeyGen**

**FrodoPKE.Enc**

**FrodoPKE.Dec**

**IND-CPA secure FrodoPKE**

**Pseudorandom A to save space**

**Basic LWE public key**
FrodoKEM construction

Algorithm 10 FrodoPKE.Enc.

Input: Message $\mu \in \mathcal{M}$ and public key $pk = (seed_A, B) \in \{0, 1\}^{\text{len}_A} \times \mathbb{Z}_q^{n \times \bar{n}}$.

Output: Ciphertext $c = (C_1, C_2) \in \mathbb{Z}_q^{\bar{m} \times n} \times \mathbb{Z}_q^{\bar{m} \times \bar{n}}$.

1: Generate $A \leftarrow \text{Frodo.Gen}(seed_A)$
2: Choose a uniformly random seed $seed_E \leftarrow \mathbb{U}(\{0, 1\}^{\text{len}_E})$
3: Sample error matrix $S' \leftarrow \text{Frodo.SampleMatrix}(seed_E, \bar{m}, n, T_X, 4)$
4: Sample error matrix $E' \leftarrow \text{Frodo.SampleMatrix}(seed_E, \bar{m}, n, T_X, 5)$
5: Sample error matrix $E'' \leftarrow \text{Frodo.SampleMatrix}(seed_E, \bar{m}, \bar{n}, T_X, 6)$
6: Compute $B' = S' A + E'$ and $V = S'B + E''$
7: return ciphertext $c \leftarrow (C_1, C_2) = (B', V + \text{Frodo.Encode}(\mu))$

- IND-CPA secure FrodoPKE
- FrodoPKE.KeyGen
- FrodoPKE.Enc
- FrodoPKE.Dec
- Basic LWE ciphertext
- Shared secret
- Key transport using public key encryption
**FrodoKEM construction**

- IND-CPA secure FrodoPKE
- FrodoPKE.KeyGen
- FrodoPKE.Enc
- FrodoPKE.Dec

---

### Algorithm 11 FrodoPKE.Dec.

**Input:** Ciphertext $c = (C_1, C_2) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times \tilde{n}}$ and secret key $sk = S \in \mathbb{Z}_q^{n \times \tilde{n}}$.

**Output:** Decrypted message $\mu' \in \mathcal{M}$.

1. Compute $M = C_2 - C_1 S$
2. **return** message $\mu' = $ Frodo.Decode($M$)
FrodoKEM construction

IND-CPA secure FrodoPKE

FrodoPKE.KeyGen

FrodoPKE.Enc

FrodoPKE.Dec

IND-CCA secure FrodoKEM

FrodoKEM.KeyGen

FrodoKEM.Encaps

FrodoKEM.Decaps

Targhi–Unruh Quantum Fujisaki–Okamoto (QFO) transform

Adds well-formedness checks Extra hash value Implicit rejection Requires negligible error rate
### FrodoKEM parameters

<table>
<thead>
<tr>
<th></th>
<th>FrodoKEM-640</th>
<th>FrodoKEM-976</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimension</strong> $n$</td>
<td>640</td>
<td>976</td>
</tr>
<tr>
<td><strong>Modulus</strong> $q$</td>
<td>$2^{15}$</td>
<td>$2^{16}$</td>
</tr>
<tr>
<td><strong>Error distribution</strong></td>
<td>Approx. Gaussian [-11, ..., 11], $\sigma = 2.75$</td>
<td>Approx. Gaussian [-10, ..., 10], $\sigma = 2.3$</td>
</tr>
<tr>
<td><strong>Failure probability</strong></td>
<td>$2^{-148}$</td>
<td>$2^{-199}$</td>
</tr>
<tr>
<td><strong>Ciphertext size</strong></td>
<td>9,736 bytes</td>
<td>15,768 bytes</td>
</tr>
<tr>
<td><strong>Estimated security</strong> (cryptanalytic)</td>
<td>$2^{143}$ classical $2^{103}$ quantum</td>
<td>$2^{209}$ classical $2^{150}$ quantum</td>
</tr>
<tr>
<td><strong>Runtime</strong></td>
<td>1.1 msec</td>
<td>2.1 msec</td>
</tr>
</tbody>
</table>
Other applications of LWE
Fully homomorphic encryption from LWE

- KeyGen(): \( s \leftarrow \chi(\mathbb{Z}_q^n) \)

- Enc\((sk, \mu \in \mathbb{Z}_2)\): Pick \( c \in \mathbb{Z}_q^n \) such that \( \langle s, c \rangle = e \mod q \) where \( e \in \mathbb{Z} \) satisfies \( e \equiv \mu \mod 2 \).

- Dec\((sk, c)\): Compute \( \langle s, c \rangle \in \mathbb{Z}_q \), represent this as \( e \in \mathbb{Z} \cap \left[ -\frac{q}{2}, \frac{q}{2} \right) \). Return \( \mu' \leftarrow e \mod 2 \).

[Brakerski, Vaikuntanathan; FOCS 2011]
Fully homomorphic encryption from LWE

\[ \mathbf{c}_1 + \mathbf{c}_2 \text{ encrypts } \mu_1 + \mu_2: \]

\[ \langle \mathbf{s}, \mathbf{c}_1 + \mathbf{c}_2 \rangle = \langle \mathbf{s}, \mathbf{c}_1 \rangle + \langle \mathbf{s}, \mathbf{c}_2 \rangle = e_1 + e_2 \mod q \]

Decryption will work as long as the error \( e_1 + e_2 \) remains below \( q/2 \).
Fully homomorphic encryption from LWE

Let \( \mathbf{c}_1 \otimes \mathbf{c}_2 = (c_{1,i} \cdot c_{2,j})_{i,j} \in \mathbb{Z}_q^{n^2} \) be the tensor product (or Kronecker product).

\( \mathbf{c}_1 \otimes \mathbf{c}_2 \) is the encryption of \( \mu_1 \mu_2 \) under secret key \( \mathbf{s} \otimes \mathbf{s} \):

\[
\langle \mathbf{s} \otimes \mathbf{s}, \mathbf{c}_1 \otimes \mathbf{c}_2 \rangle = \langle \mathbf{s}, \mathbf{c}_1 \rangle \cdot \langle \mathbf{s}, \mathbf{c}_2 \rangle = e_1 \cdot e_2 \mod q
\]

Decryption will work as long as the error \( e_1 \cdot e_2 \) remains below \( q/2 \).

[Brakerski, Vaikuntanathan; FOCS 2011]
Fully homomorphic encryption from LWE

• Error conditions mean that the number of additions and multiplications is limited.
• Multiplication increases the dimension (exponentially), so the number of multiplications is again limited.

• There are techniques to resolve both of these issues.
  • **Key switching** allows converting the dimension of a ciphertext.
  • **Modulus switching** and **bootstrapping** are used to deal with the error rate.
Digital signatures [Lyubashevsky 2011]

- KeyGen(): \( S \leftarrow \mathcal{S} \{-d, \ldots, 0, \ldots, d\}^{m \times k}, \ A \leftarrow \mathbb{Z}_q^{n \times m}, \ T \leftarrow A S. \)
  Secret key: \( S \); public key: \( (A, T) \).

- Sign(\( S, \mu \)): \( y \leftarrow \chi^m; \ c \leftarrow H(Ay, \mu); \ z \leftarrow Sc + y. \)
  With prob. \( p(z) \) output \( (z, c) \), else restart Sign. "Rejection sampling"

- Vfy((\( A, T \), \( \mu \), \( (z, c) \))): Accept iff \( \|z\| \leq \eta \sigma \sqrt{m} \) and \( c = H(Az - Tc, \mu) \)
Lattice-based signature schemes submitted to NIST

- CRYSTALS-Dilithium (MLWE)
- Falcon (NTRU)
- pqNTRUsign (NTRU)
- qTESLA (RLWE)
Post-quantum security models
Post-quantum security models

• Is the adversary quantum?

• If so, at what stage(s) in the security experiment?

• If so, can the adversary interact with honest parties (make queries) quantumly?

• If so, and if the proof is in the random oracle model, can the adversary access the random oracle quantumly?
Public key encryption security models

**IND-CCA**

- A is classical

\[ \text{Exp}^{\text{ind-cca}}_{\Pi}(A) \]

1. \((pk, sk) \leftarrow \text{KeyGen}()\)
2. \((m_0, m_1, st) \leftarrow A^{\text{Enc}(pk, \cdot), \text{Dec}(sk, \cdot)}(pk)\)
3. \(b \leftarrow \{0, 1\}\)
4. \(c^* \leftarrow \text{Enc}(pk, m_b)\)
5. \(b' \leftarrow A^{\text{Enc}(pk, \cdot), \text{Dec}(sk, \neq c^*)}(st, c^*)\)

**Quantum security models**

- "Future quantum"
  - A is quantum in line 5 but always has only classical access to Enc and Dec

- "Post-quantum"
  - A is quantum in lines 2 and 5 but always has only classical access to Enc & Dec

- "Fully quantum"
  - A is quantum in lines 2 and 5 and has quantum (superposition) access to Enc and Dec

Symmetric crypto generally quantum-resistant, unless in fully quantum security models.

[Kaplan et al., CRYPTO 2016]
Quantum random oracle model

- If the adversary is locally quantum (e.g., future quantum, post-quantum), should the adversary be able to query its random oracle quantumly?
  - No: We imagine the adversary only interacting classically with the honest system.
  - Yes: The random oracle model artificially makes the adversary interact with something (a hash function) that can implement itself in practice, so the adversary could implement it quantumly.
    - QROM seems to be prevalent these days

- Proofs in QROM often introduce tightness gap
  - QROM proofs of Fujisaki–Okamoto transform from IND-CPA PKE to IND-CCA PKE very hot topic right now
Transitioning to PQ crypto
Retroactive decryption

• A passive adversary that records today's communication can decrypt once they get a quantum computer
  • Not a problem for some scenarios
  • Is a problem for other scenarios

• How to provide potential post-quantum security to early adopters?
Hybrid ciphersuites

- Use pre-quantum and post-quantum algorithms together
- Secure if either one remains unbroken

Why hybrid?
- Potential post-quantum security for early adopters
- Maintain compliance with older standards (e.g. FIPS)
- Reduce risk from uncertainty on PQ assumptions/parameters

Need to consider backward compatibility for non-hybrid-aware systems
# Hybrid ciphersuites

<table>
<thead>
<tr>
<th>Key exchange</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>1   Hybrid traditional + PQ</td>
<td>Single traditional</td>
</tr>
<tr>
<td>2   Hybrid traditional + PQ</td>
<td>Hybrid traditional + PQ</td>
</tr>
<tr>
<td>3   Single PQ</td>
<td>Single traditional</td>
</tr>
<tr>
<td>4   Single PQ</td>
<td>Single PQ</td>
</tr>
</tbody>
</table>

Likely focus for next 10 years
Hybrid post-quantum key exchange

**TLS 1.2**

- Prototypes and software experiments:
  - Bos, Costello, Naehrig, Stebila, S&P 2015
  - Bos, Costello, Ducas, Mironov, Naehrig, Nikolaenko, Raghunathan, Stebila, ACM CCS 2016
- Google Chrome experiment
- liboqs OpenSSL fork
  - [https://openquantumsafe.org/](https://openquantumsafe.org/)
- Microsoft OpenVPN fork

**TLS 1.3**

- Prototypes:
  - liboqs OpenSSL fork
- Internet drafts:
  - Whyte et al.
  - Shank and Stebila
Hybrid signatures

**X.509 certificates**

- How to convey multiple public keys & signatures in a single certificate?
- Proposal: second certificate in X.509 extension
- Experimental study of backward compatibility

**Theory**

- Properties of different combiners for multiple signature schemes
- Hierarchy of security notions based on quantumness of adversary

[Bindel, Herath, McKague, Stebila. PQCrypto 2017]
Open Quantum Safe Project

Potential and reported uses (outside the OQS project)

Integration into forks of widely used open-source projects

C language library, common API
- x86/x64 (Linux, Mac, Windows)
- ARM (Android, Linux)
Two versions:
- master branch: high quality audited code; MIT licensed
- nist-branch: as many NIST submissions as possible

liboqs
master branch, nist-branch

key exchange / KEMs
signatures

- code-based
- hash-based
- isogenies
- lattice-based
- multi-variate quadratic

Language SDKs
- Python
- Rust
- ...

OpenSSL
- TLS 1.2
- TLS 1.3

OpenSSH

Apache httpd

OpenVPN

Summary
Summary

- Intro to post-quantum cryptography
- Learning with errors problems
  - LWE, Ring-LWE, Module-LWE, Learning with Rounding, NTRU
  - Search, decision
  - With uniform secrets, with short secrets
- Public key encryption from LWE
  - Regev
  - Lindner–Peikert
- Security of LWE
  - Lattice problems – GapSVP
- KEMs and key agreement from LWE
- Other applications of LWE
- PQ security models
- Transitioning to PQ crypto
More reading

• Post-Quantum Cryptography  
  by Bernstein, Buchmann, Dahmen

• A Decade of Lattice Cryptography  
  by Chris Peikert  
  https://web.eecs.umich.edu/~cpeikert/pubs/lattice-survey.pdf

• NIST Post-quantum Cryptography Project  
  http://nist.gov/pqcrypto