CRYSTALS (Crypto Suite for Algebraic Lattices) and Open Quantum Safe

Tancrède Lepoint
SRI International
Lattices are represented by a basis. This basis is not unique.

How did I construct this drawing?
Lattices

\[(x, y) \in \mathbb{Z}^2 \mapsto \begin{pmatrix} 1 & 1.2 \\ -0.5 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \in L\]

In cryptography:

- matrices over \( \mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z} \) for an integer \( q \)
- in this talk, this is a lattice:
Hard Problem: Learning With Errors [Regev'05]

We can take the secret to be small, the matrix to be square [ACPS'09, LPR'10].
Hard Problem: Learning With Errors [Regev '05]

We can take the secret to be small, the matrix to be square [ACPS’09, LPR’10]
Hard Problem: Learning With Errors [Regev'05]

Decision LWE: Distinguish

We can take the secret to be small, the matrix to be square [ACPS'09, LPR'10]
Key Encapsulation Mechanism in TLS

Need replacement for key exchange (and signature)

Server

- ServerHello
- Certificate
- ServerKeyExchange
- CertificateRequest
- ServerHelloDone
- Certificate
- ClientKeyExchange
- CertificateVerify
- [ChangeCipherSpec]
- ClientComputeKey
- Finished
- accept
- CertificateVerify
- [ChangeCipherSpec]
- ServerComputeKey
- Finished

Client

- ClientHello
- Certificate
- ClientKeyExchange
- CertificateVerify
- [ChangeCipherSpec]
- ClientComputeKey
- Finished
- accept
- CertificateVerify
- [ChangeCipherSpec]
- ServerComputeKey
- Finished
Key Encapsulation Mechanism in TLS

- Need replacement for key exchange (and signature)
- We can use a Key Encapsulation Mechanism
Key Encapsulation Mechanism in TLS

Need replacement for key exchange (and signature)

We can use a Key Encapsulation Mechanism

ServerKeyExchange: Generate a pk
Key Encapsulation Mechanism in TLS

- Need replacement for key exchange (and signature)
- We can use a Key Encapsulation Mechanism

- ServerKeyExchange: Generate a pk
- ClientKeyExchange: Generate a session key $k = f(x)$ and encrypts $x$
Key Encapsulation Mechanism in TLS

- Need replacement for key exchange (and signature)
- We can use a Key Encapsulation Mechanism

- ServerKeyExchange: Generate a pk
- ClientKeyExchange: Generate a session key $k = f(x)$ and encrypts $x$
- ServerComputeKey: Decrypt to recover $x$ and compute $k = f(x)$
Key Encapsulation Mechanism in TLS

- Need replacement for key exchange (and signature)
- We can use a Key Encapsulation Mechanism

\[ \text{ServerKeyExchange} : \text{Generate a pk} \]
\[ \text{ClientKeyExchange} : \text{Generate a session key } k = f(x) \text{ and encrypts } x \]
\[ \text{ServerComputeKey} : \text{Decrypt to recover } x \text{ and compute } k = f(x) \]

We want an encryption scheme :)
Regev's encryption scheme [Regev'05,GPV'08]

Key generation

Encryption

Decryption

\[ \begin{array}{ccc}
\times & + & = \\
= & \approx & -1
\end{array} \]
Using Regev's Scheme

- Dimension of $A \approx 752 \times 752$
- Dimension of $B \approx 752 \times 256$
- Modulus $q = 2^{15} = 32768$
Using Regev's Scheme

In TLS:

- Dimension of \( A \approx 752 \times 752 \)
- Dimension of \( B \approx 752 \times 256 \)
- Modulus \( q = 2^{15} = 32768 \)

C → S:
\[ 752 \times 256 \times 15 = 2.8\text{Mb} \]

S → C:
\[ (752 + 256) \times 15 = 15\text{Kb} \]
Reducing communication: Frodo (CCS'16)

- Public key: extract several bits per column (requires larger modulus) and compress

\[ \in \text{trunc}_C(\mathbb{Z}_q^{256 \times 8}) \]

(Comm.: 11,377 bytes)

- Reconciliation instead of sending back ciphertext of key, set key = $\text{LSB}_B(V)$ and send $B + 1$-th bit of $V$ (reconciliation)

  - Communication: 11,296 bytes

  - Idea from [Ding'12, Peikert'14]
Reducing communication with rings: NewHope (Usenix'16)

- Reconciliation (or not: NewHope-Simple)
- Use *ring* lattices \([LM'06,PR'06]\), i.e.,
  
  \[
  \begin{align*}
  a(x) & \equiv \sum_{i=0}^{n-1} a_i x^i \\
  s(x) & \equiv \sum_{i=0}^{n-1} s_i x^i \\
  p(x) & = a(x) \cdot s(x) \mod (x^n + 1)
  \end{align*}
  \]

- Columns: (anti-)cyclic rotations $\rightarrow$ need only to store *n* coefficients
- Polynomial representation:

- Communication: around 4KB (2kB each way)
- Successful experiment by Google from July to November 2016
Can we do better? Yes, because $1024 \gg 256$

Ring Encryption

---

In Frodo, we could select the

In NewHope, we have many

0 coefficients

increases communication

we could use error-correcting codes to correct and reduce

decryption error

9/15
Can we do better? Yes, because $1024 \gg 256$

- In Frodo, we could select the $B$ to only encrypt $K$
- In NewHope, we have many 0 coefficients
  - increases communication
  - we could use error-correcting codes to correct and reduce decryption error
CRYSTALS' Kyber (Real World Crypto 2017)
Joint work with Shi Bai, Joppe Bos, Léo Ducas, Eike Kiltz, Vadim Lyubashevsky, John M Schank, Peter Schwabe, and Damien Stehlé

Representation

\[ \mathbf{A} \times \vec{s} = \vec{b} \text{ over } \left( \mathbb{Z}_q[x]/(x^{256} + 1) \right)^4 \]
CRYSTALS' Kyber (Real World Crypto 2017)
Joint work with Shi Bai, Joppe Bos, Léo Ducas, Eike Kiltz, Vadim Lyubashevsky, John M Schank, Peter Schwabe, and Damien Stehlé

Representation

\[ \vec{s} \times \vec{a} = \vec{b} \text{ over } (\mathbb{Z}_q[x]/(x^{256} + 1))^4 \]
CRYS'TAILS' Kyber (Real World Crypto 2017)
Joint work with Shi Bai, Joppe Bos, Léo Ducas, Eike Kiltz, Vadim Lyubashevsky, John M Schank, Peter Schwabe, and Damien Stehlé

Representation

\[ \mathbf{A} \times \vec{s} = \vec{b} \text{ over } (\mathbb{Z}_q[x]/(x^{256} + 1))^4 \]

- Communication: around 2KB (1KB each side)
- No reconciliation and better security (CCA scheme, long term keys, etc.)
Developers' aspects

Building blocks:

- **Number Theoretic Transform** (dim 256)
- **Arithmetic modulo** \( q = 7681 < 2^{13} \)
- "Noise" distribution: draw \( 2\eta \) bits and compute \( \sum_{i=1}^{\eta} (a_i - b_i) \)
- Working with module lattices:

  ```c
  void polyvec_ntt(polyvec *r) {
      for(int i=0; i<3; i++) poly_ntt(&r->vec[i]);
  }
  ```

- Increasing security is easy: essentially amounts to modify 3 in 4
Open Quantum Safe
https://openquantumsafe.org

OQS bench
OpenSSL
OTR (soon)
...

Integrations

Open Quantum Safe Library

Open Quantum Safe Library

OQS-KEX
OQS-SIG

Lattices
Codes
SIDH
MQ
...

API

Primitives impl.

Primitives impl.

(soon)


```bash
./openssl speed
AWS c4.large (Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Server 0</th>
<th>Client</th>
<th>Server 1</th>
<th>Communication</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(ms)</td>
<td></td>
<td></td>
<td>S → C</td>
<td>C → S</td>
</tr>
<tr>
<td>SIDH</td>
<td>15.84</td>
<td>35.14</td>
<td>14.97</td>
<td>564</td>
<td>564</td>
</tr>
<tr>
<td>McBits</td>
<td>69.92</td>
<td>0.04</td>
<td>0.15</td>
<td>311,736</td>
<td>109</td>
</tr>
<tr>
<td>LWE Frodo</td>
<td>0.91</td>
<td>1.33</td>
<td>0.16</td>
<td>11,377</td>
<td>11,296</td>
</tr>
<tr>
<td>Ring-LWE BCNS15</td>
<td>0.72</td>
<td>1.17</td>
<td>0.16</td>
<td>4,096</td>
<td>4,224</td>
</tr>
<tr>
<td>R. NewHope</td>
<td>0.05</td>
<td>0.08</td>
<td>0.02</td>
<td>1,824</td>
<td>2,048</td>
</tr>
<tr>
<td>R. NewHope-Simple</td>
<td>1,824</td>
<td>2,176</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module-LWE Kyber</td>
<td>0.06</td>
<td>0.08</td>
<td>0.09</td>
<td>1,088</td>
<td>1,152</td>
</tr>
</tbody>
</table>
```
CRYSMTALS: Wrap Up

CRYSMTALS: Cryptographic Suite for Algebraic Lattices

- **KYBER**: CCA-KEM
  - Can be used for key exchange, and in KEM+DEM
  - Similar to NewHope (i.e., no impediment to use it)
  - Better communication and better security than NewHope
  - Nearly as efficient as NewHope
  - Easy to implement: arithmetic mod 7681, 256-NTT, sum of bits. No reconciliation or Gaussian sampling.

- **DILITHIUM**: Digital Signature
  - Same building blocks than Kyber with module lattices
    - easiness of implementation, reusability
  - Uniform error distributions
  - Fiat-Shamir paradigm
Thank You

https://github.com/pq-crystals