Post-quantum key exchange for the TLS protocol from R-LWE

Craig Costello
Microsoft Research

joint work with
Joppe Bos (NXP), Michael Naehrig (MSR), Douglas Stebila (QUT)
This work: R-LWE in TLS

- All (public-key) ciphersuites currently offered in TLS will break if a large-scale quantum computer is built.

- This work: build ciphersuites that (hopefully) won't break.

  e.g. RLWE-ECDSA-AES128-GCM-SHA256
This work: R-LWE key agreement in TLS

• In this work, we start by looking at post-quantum key agreement only

• Assumption: large-scale quantum computers don’t exist now, but what if we want to protect today’s communications against tomorrow’s adversary?

• Signatures still done with traditional primitives RSA/ECDSA (we only need authentication to be secure now)

  e.g. RLWE-ECDSA-AES128-GCM-SHA256
The learning with errors (LWE) problem

LWE problem: given blue, find red
The learning with errors (LWE) problem

LWE problem: given blue, find red
The learning with errors (LWE) problem

\[ Z_{13}^{7 \times 4} \times Z_{13}^{4 \times 1} + Z_{13}^{7 \times 1} = Z_{13}^{7 \times 1} \]

**LWE problem:** given blue, find red
Toy example versus real-world example

$Z_{13}^{7 \times 4}$

$Z_{4093}^{640 \times 256}$

$640 \times 256 \times 12 = 1966080 \text{ bits}$

$= 245 \text{ kB} !!$
The learning with errors (LWE) problem

LWE problem: given blue, find red
The **ring** learning with errors (**R-LWE**) problem

<table>
<thead>
<tr>
<th>random $Z_{13}^{7\times4}$</th>
<th>secret $Z_{13}^{4\times1}$</th>
<th>small $Z_{13}^{7\times1}$</th>
<th>ind. from random $Z_{13}^{7\times1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 11 10</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>10 4 11</td>
<td>9</td>
<td>-1</td>
<td>6</td>
</tr>
<tr>
<td>11 10 4 1</td>
<td>11</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1 11 10 4</td>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12 7 3 4</td>
<td></td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>4 12 7 3</td>
<td></td>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>3 4 12 7</td>
<td></td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
The **ring** learning with errors (**R-LWE**) problem

\[
\begin{align*}
\text{random} & \quad Z_{13}^{7 \times 4} \\
\begin{array}{cccc}
4 & 1 & 11 & 10 \\
3 & 4 & 1 & 11 \\
2 & 3 & 4 & 1 \\
12 & 2 & 3 & 4 \\
12 & 7 & 3 & 4 \\
9 & 12 & 7 & 3 \\
10 & 9 & 12 & 7 \\
\end{array} & \quad \times \quad \text{secret} & \quad Z_{13}^{4 \times 1} \\
\begin{array}{c}
6 \\
9 \\
11 \\
\end{array} & \quad + \quad \text{small} & \quad Z_{13}^{7 \times 1} \\
\begin{array}{c}
0 \\
-1 \\
1 \\
\end{array} & \quad \text{ind. from random} & \quad Z_{13}^{7 \times 1} \\
\begin{array}{c}
4 \\
3 \\
4 \\
\end{array}
\end{align*}
\]

\[
\begin{array}{cccc}
4 & 1 & 11 & 10 \\
3 & 4 & 1 & 11 \\
2 & 3 & 4 & 1 \\
12 & 2 & 3 & 4 \\
12 & 7 & 3 & 4 \\
9 & 12 & 7 & 3 \\
10 & 9 & 12 & 7 \\
\end{array} \times \begin{array}{c}
6 \\
9 \\
11 \\
\end{array} + \begin{array}{c}
0 \\
-1 \\
1 \\
\end{array} = \begin{array}{c}
4 \\
3 \\
4 \\
\end{array}
\]
The *ring* learning with errors (R-LWE) problem

\[
\begin{bmatrix}
4 & 1 & 11 & 10 \\
3 & 4 & 1 & 11 \\
2 & 3 & 4 & 1 \\
12 & 2 & 3 & 4 \\
\end{bmatrix}
\times
\begin{bmatrix}
6 \\
9 \\
11 \\
11 \\
\end{bmatrix}
+ \begin{bmatrix}
0 \\
-1 \\
1 \\
1 \\
\end{bmatrix}
= \begin{bmatrix}
4 \\
3 \\
4 \\
12 \\
\end{bmatrix}
\]

LWE problem: given blue, find red
The **ring** learning with errors (R-LWE) problem

\[ Z_{13}^{4 \times 4} \rightarrow Z_{13}[x]/(x^4 + 1) \]

\[
\begin{array}{cccc}
4 & 1 & 11 & 10 \\
3 & 4 & 1 & 11 \\
2 & 3 & 4 & 1 \\
12 & 2 & 3 & 4 \\
\end{array}
\]

\[ = x \cdot (4 + 1x + 11x^2 + 10x^3) \]

\[ = x^2 \cdot (4 + 1x + 11x^2 + 10x^3) \]

\[ = x^3 \cdot (4 + 1x + 11x^2 + 10x^3) \]
The **ring** learning with errors (**R-LWE**) problem

\[
\begin{array}{c}
4 + 1x + 11x^2 + 10x^3 \\
\times 6 + 9x + 11x^2 + 11x^3 \\
+ 0 - 1x + 1x^2 + 1x^3 \\
\hline
10 + 5x + 10x^2 + 7x^3
\end{array}
\]

\[
\frac{Z_{13}[x]}{\langle x^4 + 1 \rangle}
\]

**R-LWE problem:** given **blue**, find **red**
The ring learning with errors (R-LWE) problem

\[ 4 + 1x + 11x^2 + 10x^3 \]
\[ \times 1 + 0x - 1x^2 - 1x^3 \]
\[ + 0 - 1x + 1x^2 + 1x^3 \]
\[ \frac{Z_{13}[x]}{\langle x^4 + 1 \rangle} \]

\[ \frac{3 + 8x + 5x^2 + 6x^3}{\langle x^4 + 1 \rangle} \]

R-LWE problem (small secrets): given blue, find (small!) red
The ring learning with errors (R-LWE) problem (the 128-bit secure version)

\[
\begin{align*}
2792930407 + \cdots + 2938465015x^{1023} \\
\times & \quad 5 - 3x \ldots + 9x^{1022} - 1x^{1023} \\
\times & \quad 2 + 4x \ldots - 0x^{1022} + 6x^{1023} \\
\end{align*}
\frac{Z_{2^{32} - 1}[x]}{\langle x^{1024} + 1 \rangle}
\]

\[
3159804584 + \cdots + 1153769078x^{1023}
\]

R-LWE problem: given blue, find (small!) red
R-LWE-DH: key agreement in $R_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$

**public:** “big” $a \in R_q$

**secret:** “small” $e, s \in R_q$

$$a \cdot s + e$$

$$(s \cdot (a \cdot s' + e')) \approx s \cdot a \cdot s'$$

**public:** “big” $a \in R_q$

**secret:** “small” $e', s' \in R_q$

$$a \cdot s' + e'$$

$$(s' \cdot (a \cdot s + e)) \approx s \cdot a \cdot s'$$
Approximate agreement mod $q$

4079331841 + 1894732145 \cdot x + \cdots + 472608255 \cdot x^{1022} + 516748383 \cdot x^{1023} \\
\\
4079332556 + 1894733033 \cdot x + \cdots + 472607765 \cdot x^{1022} + 516748363 \cdot x^{1023} \\
\\
ROUND 0 1 0 0

This will work most of the time (fails $\approx 1/2^{10}$), but we need **exact agreement** i.e. what happens if one of the coefficients is in the “danger zone(s)”
Making approximate agreement exact in $\mathbb{Z}_q$

If $u - v < q/8$ (i.e., always!), two values $u, v \in \mathbb{Z}_q$ will agree so long as $|u - v| < q/8$ (i.e., always!).

R-LWE-DH: exact key agreement

**public:** “big” $a \in R_q$

**secret:** “small” $e, s \in R_q$

$a \cdot s + e$

$a \cdot s' + e'$ and $\mathcal{\{\green{\mathcal{\{}}}, \red{\mathcal{\{}}}}^n \in \{0,1\}^n$

**RECONCILE**($s \cdot (a \cdot s' + e'), \mathcal{\{\green{\mathcal{\{}}}, \red{\mathcal{\{}}}}^n)$ $\equiv$ **ROUND**($s' \cdot (a \cdot s + e)$)

both parties now share $k \in \{0,1\}^n$
A secure key agreement protocol

- Prove that scheme is secure under “Exact DDH-like problem”
- Show that “Exact DDH-like problem” is hard if decision R-LWE problem is

Secure integration into the TLS

- Integrate R-LWE key agreement into the TLS protocol
- Use Jager et al. “Authenticated and confidential channel establishment” (ACCE) model (Crypto2012)
- Prove that “TLS-signed R-LWE is a secure ACCE”
Implementation aspect 1: polynomial arithmetic

• Polynomial multiplication in $R_q = \mathbb{Z}_q[x]/\langle x^{1024} + 1 \rangle$ done with Nussbaumer’s FFT ($2^l = r \cdot k$)

\[
\frac{R[X]}{\langle X^{2^l} + 1 \rangle} \equiv \frac{\left( \frac{R[Z]}{\langle Z^r + 1 \rangle} \right)}{\langle X^k - Z \rangle}
\]

• Rather than working modulo degree-1024 polynomial with coefficients in $\mathbb{Z}_q$, work modulo:
  - degree-256 polynomial whose coefficients are themselves polynomials modulo a degree-4 polynomial, or
  - degree-32 polynomials whose coefficients are polynomials modulo degree-8 polynomials whose coefficients are polynomials ...
Implementation aspect 2: sampling discrete Gaussians

$$D_{Z,\sigma}(x) = \frac{1}{S} e^{\frac{-x^2}{2\sigma^2}} \text{ for } x \in Z$$

(for us: $\sigma \approx 3.2, S = 8$)

- Security (proofs) require “small” elements to be within statistical distance $2^{-128}$ of true discrete Gaussian $D_{Z,\sigma}(x)$

- Inversion sampling: precompute table of cumulative probabilities
  (for us: 52 elements of 192-bits in size: $\approx 10,000$ bits)

- Each coefficient requires six 192-bit integer comparisons (51 if “constant-time”), and there are 1024 coefficients!!!
The price of post-quantum paranoia, part I

<table>
<thead>
<tr>
<th>Operation</th>
<th>Constant-time</th>
<th>Cycles non-constant-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample ( \xleftarrow{$} \chi )</td>
<td>1 042 700</td>
<td>668 000</td>
</tr>
<tr>
<td>FFT multiplication</td>
<td>342 800</td>
<td>—</td>
</tr>
<tr>
<td>FFT addition</td>
<td>1 660</td>
<td>—</td>
</tr>
<tr>
<td>dbl(\cdot) and crossrounding ( \langle \cdot \rangle_{2q,2} )</td>
<td>23 500</td>
<td>21 300</td>
</tr>
<tr>
<td>rounding ( \lfloor \cdot \rfloor_{2q,2} )</td>
<td>5 500</td>
<td>3 700</td>
</tr>
<tr>
<td>reconciliation ( \text{rec}(\cdot, \cdot) )</td>
<td>14 400</td>
<td>6 800</td>
</tr>
</tbody>
</table>

(Inel Core i5 (4570R) @ 2.7GHz)
Table 2: Average runtime in milliseconds of cryptographic operations using `openssl1` speed

<table>
<thead>
<tr>
<th>Operation</th>
<th>Client constant-time</th>
<th>Client non-constant-time</th>
<th>Server constant-time</th>
<th>Server non-constant-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-LWE key generation</td>
<td>0.9</td>
<td>1.7</td>
<td>0.6</td>
<td>1.3</td>
</tr>
<tr>
<td>R-LWE Bob shared secret</td>
<td>0.5</td>
<td>(1.1)</td>
<td>0.4</td>
<td>(0.9)</td>
</tr>
<tr>
<td>R-LWE Alice shared secret</td>
<td>(0.1)</td>
<td>0.4</td>
<td>(0.1)</td>
<td>0.4</td>
</tr>
<tr>
<td><strong>Total R-LWE runtime</strong></td>
<td><strong>1.4</strong></td>
<td><strong>2.1</strong></td>
<td><strong>1.0</strong></td>
<td><strong>1.7</strong></td>
</tr>
<tr>
<td>EC point multiplication, nistp256</td>
<td>0.4</td>
<td>0.7</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td><strong>Total ECDH runtime</strong></td>
<td><strong>0.8</strong></td>
<td><strong>1.4</strong></td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RSA sign, 3072-bit key</td>
<td>(3.7)</td>
<td>8.8</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RSA verify, 3072-bit key</td>
<td>0.1</td>
<td>(0.2)</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Numbers in parentheses are reported for completeness, but do not contribute to the runtime in the client and server’s role in the TLS protocol.
The price of post-quantum paranoia, part III

![Graph showing performance of HTTPS using Apache with OpenSSL]

Table 3: Performance of HTTPS using Apache with OpenSSL

<table>
<thead>
<tr>
<th></th>
<th>ECDHE</th>
<th>RSA</th>
<th>RLWE</th>
<th>ECDSA</th>
<th>RSA</th>
<th>HYBRID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections / second</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 B payload</td>
<td>645.9</td>
<td>177.4</td>
<td>507.5</td>
<td>164.2</td>
<td>362.9</td>
<td>145.1</td>
</tr>
<tr>
<td>1 KiB payload</td>
<td>641.6</td>
<td>177.0</td>
<td>505.9</td>
<td>163.8</td>
<td>361.0</td>
<td>145.0</td>
</tr>
<tr>
<td>10 KiB payload</td>
<td>630.2</td>
<td>176.2</td>
<td>494.9</td>
<td>161.9</td>
<td>356.2</td>
<td>144.1</td>
</tr>
<tr>
<td>100 KiB payload</td>
<td>487.6</td>
<td>161.2</td>
<td>397.6</td>
<td>150.2</td>
<td>300.5</td>
<td>134.3</td>
</tr>
<tr>
<td>Connection time (ms)</td>
<td>6.0</td>
<td>14.0</td>
<td>45.6</td>
<td>54.0</td>
<td>47.2</td>
<td>54.6</td>
</tr>
<tr>
<td>Handshake (bytes)</td>
<td>1278</td>
<td>2360</td>
<td>9469</td>
<td>10479</td>
<td>9607</td>
<td>10690</td>
</tr>
</tbody>
</table>
Summary and future work

• If you want to protect today’s secrets against tomorrow’s quantum adversary, use

\[ \text{RLWE-ECDSA-AES128-GCM-SHA256} \]

in TLS for a small overhead

• Future work, part II: protecting tomorrow’s secrets too!

\[ \text{RLWE-RLWE-AES128-GCM-SHA256} \]
\[ \text{LWE-LWE-AES128-GCM-SHA256} \]
\[ \text{????-????-AES128-GCM-SHA256} \]

• Future work, part I: a tonne of unexplored optimizations (this is our first go)
  - e.g: we didn’t do assembly/precomputation/parallelizing
  - e.g: alternative FFT’s
  - e.g: much faster/compact sampling algorithms likely
The paper (to appear at Oakland S&P)

Magma code:

C code integrated into OpenSSL:
https://github.com/dstebila/rlwekex