A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem

Jintai Ding

University of Cincinnati
This includes joint work with X. Ling, X. Xiang, J. Zhang, Z. Zhang, M. Snook, O. Dagdelen

Oct. 7, 2014
Practical Challenge

- Quantum computing will break many public-key cryptographic algorithms/schemes
  - Key agreement (e.g. DH and MQV)
  - Digital signatures (e.g. RSA and DSA)
  - Encryption (e.g. RSA)

- These algorithms have been used to protect Internet protocols (e.g. IPsec) and applications (e.g. TLS)

- NIST is studying “quantum-safe” replacements

- This talk will focus on practical aspects
  - For security, see Yi–Kai Liu’s talk later today
Where do we really need public key cryptosystems?

- Digital signature – authentication
  - Software update
- Public key encryption systems are almost never used to send information but keys — **key agreement**
  - SSL TLS
- We can achieve this goal with encryption or **key exchange** like Diffie-Hellmann.
Encryption (Key Transport): Party A uses Party B’s public key to encrypt a random string and sends the ciphertext to B. B decrypts it and get the random string.

In practice, public key encryption is only used to transmit random keys. (The key is only determined by one party)

Using PKE can not guarantee forward security.

If the attacker gets the static secret key, then he will learn every communication made before.

The Heartbleed problem.
What’s Key Exchange

Two parties get a shared secret key over an unsecure channel.
What’s Key Exchange

Two parties get a shared secret key over an unsecure channel.
Two parties get a shared secret key over an unsecure channel.
The Elegant Diffie-Hellman Protocol

\[ g^a \leftrightarrow g^b \]

Using the simple and elegant fact:

\[ g^{ab} = (g^b)^a = (g^a)^b \].
Using the simple and elegant fact:

\[ g^{ab} = (g^b)^a = (g^a)^b. \]
Motivation and Results

Motivation:

- Can we get a DH analogy from other mathematical tools?
- Can we get KE from lattices (say, LWE, which is apparent resistance to quantum attacks)?
- If so, can we get better efficiency and better security guarantees.

Our Results:

- An efficient (2-round) key exchange protocol from LWE and RLWE.
- It is provably secure and very efficient.
Motivation and Results

Motivation:

▶ Can we get a DH analogy from other mathematical tools?
▶ Can we get KE from lattices (say, LWE, which is apparent resistance to quantum attacks)?
▶ If so, can we get better efficiency and better security guarantees.

Our Results:

▶ An Efficient (2-round) key exchange protocol from LWE and RLWE.
▶ It is provably secure and it is very efficient.
Learning with Errors (LWE) [Oded Regev 2005]

Goal: **distinguishing** “noisy inner products” from uniform.

\[
\begin{align*}
\mathbf{a}_1 &\leftarrow \mathbb{Z}_q^n; \\
\mathbf{b}_1 &= \langle \mathbf{a}_1, \mathbf{s} \rangle + e_1 \mod q \\
\mathbf{a}_2 &\leftarrow \mathbb{Z}_q^n; \\
\mathbf{b}_2 &= \langle \mathbf{a}_2, \mathbf{s} \rangle + e_2 \mod q \\
\vdots &
\mathbf{a}_m &\leftarrow \mathbb{Z}_q^n; \\
\mathbf{b}_m &= \langle \mathbf{a}_m, \mathbf{s} \rangle + e_m \mod q
\end{align*}
\]

In a matrix form

\[
(A, As + e) \approx_c (A, b)
\]

Where \( \mathbf{s} \leftarrow \mathbb{Z}_q^n \), \( m = \text{poly}(n) \), \( q = \text{poly}(n) \) and \( e_i \leftarrow \chi \) is some distribution in \( \mathbb{Z} \). \( e_i \) has small size, much smaller than \( q \).
Theorem (Informal)[Reg’05]

Let $\chi$ be a discrete Gaussian distribution with parameter $0 < \alpha < 1$, s.t. $\alpha q \geq 2\sqrt{n}$. If there exists a polynomial time algorithm solves LWE problem, then there exists a quantum algorithm solves $(n/\alpha)$-SVP problems for all $n$-dimension lattices.

$\blacktriangleright$ $s \leftarrow \chi^n$ is as hard as standard LWE ($s \leftarrow \mathbb{Z}_q^n$) [ACPS’09].
Our Protocol (basic idea)

Public Parameter: $M \leftarrow \mathbb{Z}_q^{n \times n}$

\[
\begin{align*}
p_A &= Ms_A + 2e_A \\
p_B &= M^T s_B + 2e_B
\end{align*}
\]

Note that $s_A, s_B, e_A, e_B$ are "small".

The difference between $p_A s_B$ and $p_B s_A$ is even.
Our Protocol (basic idea)

Public Parameter: $M \leftarrow \mathbb{Z}_q^{n \times n}$

\[
\begin{align*}
p_A &= Ms_A + 2e_A \\
p_B &= M^T s_B + 2e_B
\end{align*}
\]

\[
\begin{align*}
s_A^T p_B &= s_A^T M^T s_B + 2s_A^T e_B \\ &\approx s_A^T M^T s_B + 2e_A^T s_B = p_A^T s_B.
\end{align*}
\]

- note that $s_A, s_B, e_A, e_B$ are “small”.
- the difference between $s_A^T p_B$ and $p_A^T s_B$ is even
Our Protocol (basic idea)

Public Parameter: $M \leftarrow \mathbb{Z}_q^{n \times n}$

$$\begin{align*}
    p_A &= Ms_A + 2e_A \\
    p_B &= M^Ts_B + 2e_B
\end{align*}$$

$\Rightarrow s_A^T p_B \approx s_A^T M^T s_B + 2e_A^T s_B = p_A^T s_B.$

- note that $s_A, s_B, e_A, e_B$ are “small”.
- the difference between $s_A^T p_B$ and $p_A^T s_B$ is even
Our Robust Modular Extractor

We first define two functions: for $q > 2$ is prime

$$\sigma_0(x) = \begin{cases} 0, & x \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor]; \\ 1, & \text{otherwise.} \end{cases} \quad \sigma_1(x) = \begin{cases} 0, & x \in [-\lfloor \frac{q}{4} \rfloor + 1, \lfloor \frac{q}{4} \rfloor + 1]; \\ 1, & \text{otherwise.} \end{cases}$$

The hint algorithm $S(y)$: $b \xleftarrow{} \{0, 1\}, S(y) = \sigma_b(y)$.

The robust extractor $E(x, \sigma)$:

$$E(x, \sigma) = \left( x + \sigma \cdot \frac{q - 1}{2} \mod q \right) \mod 2$$
Removing the Approximation

Public Parameter: \( M \leftarrow \mathbb{Z}^{n \times n}_q \)

\[ \begin{align*}
A & \quad \mathbf{p}_A \\
\mathbf{p}_B, \sigma & \leftarrow S\left(\mathbf{p}_A^T \mathbf{s}_B\right) \\
B & \quad E\left(\mathbf{p}_A^T \mathbf{s}_B, \sigma\right)
\end{align*} \]
Removing the Approximation

Public Parameter: \( M \leftarrow \mathbb{Z}_q^{n \times n} \)

\[ \begin{align*} \mathbf{p}_A & \quad \rightarrow \quad \mathbf{p}_B, \sigma \leftarrow S(\mathbf{p}_A^T \mathbf{s}_B) \\ \mathbf{p}_A & \quad \leftarrow \quad \mathbf{p}_B, \sigma \leftarrow S(\mathbf{p}_A^T \mathbf{s}_B) \end{align*} \]

- A outputs \( E(s_A^T \mathbf{p}_B, \sigma) \)
- B outputs \( E(p_A^T \mathbf{s}_B, \sigma) \)
The security proof is given from a series of hybrid experiments.

Next — the problem of authentication — man in the middle attack!!!

We can build an authenticated key exchange (AKE) protocol, which can be seen as an HMQV-like AKE from lattices.

The protocol is simple since it does not involve any other cryptographic primitives to achieve authentication (e.g., signatures) and the system is also very efficient.
AKE from ring-LWE

Party $i$

Public Key: $p_i = as_i + 2e_i \in R_q$
Secret Key: $s_i \in R_q$
where $s_i, e_i \leftarrow \chi_\alpha$

Party $j$

Public Key: $p_j = as_j + 2e_j \in R_q$
Secret Key: $s_j \in R_q$
where $s_j, e_j \leftarrow \chi_\alpha$
AKE from ring-LWE

Party $i$

Public Key: $p_i = as_i + 2e_i \in R_q$
Secret Key: $s_i \in R_q$
where $s_i, e_i \leftarrow_r \chi_\alpha$

$x_i = ar_i + 2f_i \in R_q$
where $r_i, f_i \leftarrow_r \chi_\beta$

---

Party $j$

Public Key: $p_j = as_j + 2e_j \in R_q$
Secret Key: $s_j \in R_q$
where $s_j, e_j \leftarrow_r \chi_\alpha$

$x_i$
AKE from ring-LWE

**Party i**

Public Key: \( p_i = a s_i + 2e_i \in \mathbb{R}_q \)

Secret Key: \( s_i \in \mathbb{R}_q \)

where \( s_i, e_i \leftarrow r \chi_\alpha \)

\[
x_i = a r_i + 2f_i \in \mathbb{R}_q
\]

where \( r_i, f_i \leftarrow r \chi_\beta \)

**Party j**

Public Key: \( p_j = a s_j + 2e_j \in \mathbb{R}_q \)

Secret Key: \( s_j \in \mathbb{R}_q \)

where \( s_j, e_j \leftarrow r \chi_\alpha \)

\[
y_j = a r_j + 2f_j \in \mathbb{R}_q
\]

\[
k_j = (p_i c + x_i)(s_j d + r_j) + 2g_j
\]

where \( r_j, f_j, g_j \leftarrow r \chi_\beta \)

\[
w_j = \text{Cha}(k_j) \in \{0, 1\}^n
\]

\[
\sigma_j = \text{Mod}_2(k_j, w_j) \in \{0, 1\}^n
\]

\[
sk_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)
\]

\[
c = H_1(i, j, x_i) \in R, d = H_1(j, i, y_j, x_i) \in R
\]
AKE from ring-LWE

**Party i**

Public Key: $p_i = a s_i + 2 e_i \in R_q$

Secret Key: $s_i \in R_q$

where $s_i, e_i \leftarrow_r \chi_\alpha$

- $x_i = a r_i + 2 f_i \in R_q$
  where $r_i, f_i \leftarrow_r \chi_\beta$

- $k_i = (p_j d + y_j)(s_i c + r_i) + 2 g_i$
  where $g_i \leftarrow_r \chi_\beta$

- $\sigma_i = \text{Mod}_2(k_i, w_j) \in \{0,1\}^n$

- $sk_i = H_2(i, j, x_i, y_j, w_j, \sigma_i)$

**Party j**

Public Key: $p_j = a s_j + 2 e_j \in R_q$

Secret Key: $s_j \in R_q$

where $s_j, e_j \leftarrow_r \chi_\alpha$

- $y_j = a r_j + 2 f_j \in R_q$

- $k_j = (p_i c + x_i)(s_j d + r_j) + 2 g_j$
  where $r_j, f_j, g_j \leftarrow_r \chi_\beta$

- $w_j = \text{Cha}(k_j) \in \{0,1\}^n$

- $\sigma_j = \text{Mod}_2(k_j, w_j) \in \{0,1\}^n$

- $sk_j = H_2(i, j, x_i, y_j, w_j, \sigma_j)$

$c = H_1(i, j, x_i) \in R, d = H_1(j, i, y_j, x_i) \in R$
Intuition for Security:

- We can prove the security of the system
Intuition for Security:

1. We can prove the security of the system
2. We can prove the forward security of the system
Intuition for Security:

1. We can prove the security of the system
2. We can prove the forward security of the system
3. We did preliminary implementation and it is very efficient.
Intuition for Security:

1. We can prove the security of the system
2. We can prove the forward security of the system
3. We did preliminary implementation and it is very efficient.

Parameters for implementation:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>n</th>
<th>Security (expt.)</th>
<th>$\alpha$</th>
<th>$\gamma$</th>
<th>$\log \frac{q}{\alpha}$</th>
<th>$\log q$ (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I*</td>
<td>1024</td>
<td>80 bits</td>
<td>3.397</td>
<td>101.919</td>
<td>8.5</td>
<td>40</td>
</tr>
<tr>
<td>II</td>
<td>2048</td>
<td>80 bits</td>
<td>3.397</td>
<td>161.371</td>
<td>27</td>
<td>78</td>
</tr>
<tr>
<td>III</td>
<td>2048</td>
<td>128 bits</td>
<td>3.397</td>
<td>161.371</td>
<td>19</td>
<td>63</td>
</tr>
<tr>
<td>IV</td>
<td>4096</td>
<td>128 bits</td>
<td>3.397</td>
<td>256.495</td>
<td>50</td>
<td>125</td>
</tr>
<tr>
<td>V</td>
<td>4096</td>
<td>192 bits</td>
<td>3.397</td>
<td>256.495</td>
<td>36</td>
<td>97</td>
</tr>
<tr>
<td>VI</td>
<td>4096</td>
<td>256 bits</td>
<td>3.397</td>
<td>256.495</td>
<td>28</td>
<td>81</td>
</tr>
</tbody>
</table>
 Communication Overheads:

<table>
<thead>
<tr>
<th>Choice of Parameters</th>
<th>Size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pk</td>
</tr>
<tr>
<td>I*</td>
<td>5 KB</td>
</tr>
<tr>
<td>II</td>
<td>19.5 KB</td>
</tr>
<tr>
<td>III</td>
<td>15.75 KB</td>
</tr>
<tr>
<td>IV</td>
<td>62.5 KB</td>
</tr>
<tr>
<td>V</td>
<td>48.5 KB</td>
</tr>
<tr>
<td>VI</td>
<td>40.5 KB</td>
</tr>
</tbody>
</table>

The bound $6\alpha$ with $\text{erfc}(6) \approx 2^{-55}$ is used to estimate the size of secret keys.
Timings:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initiation</th>
<th>Response</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>3.22 ms (0.02 ms)</td>
<td>8.50 ms (4.69 ms)</td>
<td>5.23 ms (4.73 ms)</td>
</tr>
<tr>
<td>II</td>
<td>12.00 ms (0.04 ms)</td>
<td>29.33 ms (14.64 ms)</td>
<td>17.28 ms (14.61 ms)</td>
</tr>
<tr>
<td>III</td>
<td>10.33 ms (0.04 ms)</td>
<td>25.83 ms (13.46 ms)</td>
<td>15.58 ms (13.40 ms)</td>
</tr>
<tr>
<td>IV</td>
<td>83.61 ms (0.08 ms)</td>
<td>156.58 ms (39.86 ms)</td>
<td>73.11 ms (39.73 ms)</td>
</tr>
<tr>
<td>V</td>
<td>61.74 ms (0.08 ms)</td>
<td>117.81 ms (32.58 ms)</td>
<td>55.64 ms (32.20 ms)</td>
</tr>
<tr>
<td>VI</td>
<td>25.42 ms (0.08 ms)</td>
<td>62.31 ms (31.32 ms)</td>
<td>36.80 ms (31.29 ms)</td>
</tr>
</tbody>
</table>

Table: Timings of Proof-of-Concept Implementations in ms (The figures in the parentheses indicate the timings with pre-computing. For comparison, by simply using the “speed” command in openssl on the same machine, the timing for dsa1024 signing algorithm is about 0.7 ms, and for dsa2048 is about 2.3 ms).

We believe our systems are very suitable for practical applications and they have very strong security.
Summary

- We build KE and AKE based on LWE and RLWE.
- They are provably secure against both classical and quantum attacks.
- We can prove the Forward Security of the AKE.
- Our preliminary implementations are very efficient.
- Our KE and AKE are strong candidates for quantum-safe crypto.
Thank You!