CRIPTANALYSIS OF COMPACT-LWE

Jonathan Bootle, Mehdi Tibouchi, Keita Xagawa
Background Information

- Lattice-based cryptographic assumption

Compact-LWE

Based on the learning-with-errors (LWE) assumption

Hoped to achieve security for smaller parameters
Background Information

- Proposed by Liu, Li, Kim, and Nepal at ACISP’17 invited talk
- Gives lightweight encryption scheme for constrained devices
Background Information

Basic Decryption Attack

①

Equivalent Secret Keys

②

Parameter Choice

③

Honest Decryption: 500 ciphertexts per second
Our Decryption: 18,000 ciphertexts per second
Background Information

Cryptanalysed by us and others
BACKGROUND
Lattices

An $n$-dimensional lattice $\mathcal{L}$ is

- A discrete additive subgroup of $\mathbb{R}^n$
- Generated by a basis $\mathcal{B} = \{b_1, \ldots, b_n\}$
- $\mathcal{L} = \sum_{i=1}^{n}(\mathbb{Z} \cdot b_i)$
Lattices

- Solve lattice problems by finding short vectors
- Example reduction algorithms are LLL and BKZ
- Add and subtract rows
- Find short basis vectors

\[
\begin{pmatrix}
  b_1 \\
  b_2
\end{pmatrix} \rightarrow \begin{pmatrix}
  b'_1 \\
  b'_2
\end{pmatrix}
\]
Learning with Errors

\[ b_i = \langle a_i, s \rangle + e_i \]
\[ b = As + e \]

Samples \( b_i \) in \( \mathbb{Z}_q \)

Uniformly random vectors \( a_i \in \mathbb{Z}_q^n \)

Secret \( s \in \mathbb{Z}_q^n \)

Noise values \( e_i \leftarrow \chi \)
Learning with Errors

Decision: does \((\mathbf{b}, A)\) look random?
Search: given \((\mathbf{b}, A)\), find \(s\)

\[
\mathbf{b}_i = \mathbf{a}_i s + \mathbf{e}_i.
\]

\(\mathbf{b}\) and \(A\) are public
\(s\) and \(e\) are private

Uniformly random vectors
\(s \in \mathbb{Z}_q^n\)
\(\mathbf{a}_i \in \mathbb{Z}_q^n\)
Learning with Errors

Decision: does \((\mathbf{b}, A)\) look random?

Search: given \((\mathbf{b}, A)\), find \(s\)

- **Modulus** \(q\)
- **Dimension** \(n\) secrets
- Noise distribution \(\chi\)
- \(e_i \sim \chi\)
- Uniformly random vectors \(\mathbf{a}_i \in \mathbb{Z}_q^n\)
- \(b_i \in \mathbb{Z}_q^n\)
- \(\mathbf{b} = \mathbf{A}s + e\)
Compact-LWE

\[ b_i = \langle a_i, s \rangle + sk_q^{-1} \cdot p \cdot e_i \]

Samples \( b_i \) in \( \mathbb{Z}_q \)

Uniformly random vectors \( a_i \in [0, b]^n \)

Secret \( s \in \mathbb{Z}_q^n \)

Noise values \( e_i \leftarrow [0, r] \)
Compact-LWE

Decision: does \((b, A)\) look random?
Search: given \((b, A)\), find \(s\)

\(b\) and \(A\) are public

\(b_i = a_i \cdot s + sk \cdot q - 1 \cdot p \cdot e_i \rightarrow [0, r]\)

\(s \in \mathbb{Z}_q^n\)

Samples \(b_i\) in \(\mathbb{Z}_q^n\)

Secret scaling factor

\(s, e\) and the scaling factor are private
Compact-LWE

\[ b_i = \langle a_i, s \rangle + skq^{-1} \cdot p \]

Scaling factor ingredients

- Noise values \( e_i \leftarrow [0, r] \)
- Secret scaling
- Noise bound \( r \)

- \( m \) samples
- Maximum size of \( A \) elements
- \( n \) secrets modulo \( q \)
- Dimension

Uniformly random vectors \( a_i \in [0, b] \)
Parameters

Public Parameters

- \( pp = (q, n, m, t, w, b) \)
- \( t \), maximum plaintext size
- \( w \), knapsack weight for encryption
- \( PK = (A, b) \)

Secret Parameters

- \( K = (s, sk, r, p) \)

\['n + 1 < m < n^2\]
\['2b(b \log_2 b + 1) < q\]
\['2\log_2 b < n\]
\['t \leq p\]
\['sk \cdot (t - 1) + wrp < q\]
\['b < r\]
Encryption Idea

- **PK** contains random-looking samples \((a_i, b_i)\) from \((A, b)\)
- Add knapsack of \(b_i\) to hide message
- Include same knapsack of \(a_i\) to allow decryption

Encrypion of \((PK, v)\):
- Randomly pick \(w\) samples \((a_{ij}, b_{ij})\) from \(PK\)
- \((a, b) = \sum_{j=1}^{w} (a_{ij}, b_{ij})\)
- Return \(c = (a, v - b)\)
Comparison of Parameters

**Compact-LWE Parameters**
- Claims 138-bit security
- $q = 2^{32}$
- $n = 13$
- $m = 74$
- $t = 2^{16}, w = 86, b = 16$

**Lizard, Classical Parameters, 2016**
- Claims 128-bit security
- $q \approx 2^{10}$
- $n = 544$
- $m = 840$
Implementation Results

- Implemented on MTM-CM5000-MSP device
- Contiki OS
- 50 encryptions per second
- **500 decryptions per second**
BASIC DECRIPTION ATTACK
Attack Strategy

- \( c = (a, v - b) = (a, b') \)
- \( (a, b) = \sum_{j=1}^{w} (a_{i_j}, b_{i_j}) \)
- Create lattice encoding knapsack
- Find a short vector with lattice reduction

\[
\begin{pmatrix}
1 & 0 & 0 & \nu \\
0 & t I_m & \kappa a & b' \\
0 & 0 & -\kappa A & b \\
\end{pmatrix}
\]

Solves knapsack
Recovers plaintext
Experimental Results

- Correctly decrypted 9998/10,000 random ciphertexts
- Roughly 16 decryptions per second
- 3.4 GHz Core i7-3770 desktop
- Sagemath, LLL in fplll

- Honest decryption: 500 decryptions per second, constrained device

- One lattice reduction per ciphertext
- Relies on low dimension n = 13
SECRET KEY RECOVERY

*equivalent secret key
Attack Strategy

\[ b_i = \langle a_i, s \rangle + sk_{q}^{-1} \cdot p \cdot e_i \]

1. Recover scaling factor using lattice reduction

2. Find other secret values by brute force

3. Compute equivalent secret using lattice reduction
Step 1: Scale-factor Recovery

- \( b = As + ke \)
- Compute short \( U \) such that \( U^T A = 0 \mod q \)
- \( Ub = k \mod q \)

Short vector in \( (Ub)^T \mod qI \)
Step 2: Recovering Secret Key Parameters

- Secret scale-factor is $k = sk_q^{-1} \cdot p$
- Brute force search for $sk$ and $p$
- Use the values which maximise $r$

$$sk \cdot (t - 1) + wrp < q$$
Step 3: Find an Equivalent Secret

- Secret is a short lattice vector
- Use with modified decryption algorithm

\[
\begin{pmatrix}
A^T & 0 \\
qI_m & 0 \\
k^{-1} & t
\end{pmatrix}
\]
Experimental Results

- Correctly decrypted 10,000/10,000 random ciphertexts
- 1.28 seconds to get a key
- 53 microseconds per ciphertext
- Over 18,000 decryptions per second
PARAMETER CHOICE
Hardness Reductions

Breaking Compact-LWE

[LLKN17]

Breaking LWE

Dimension 13

Easy...
Hardness Reductions

Breaking Compact-LWE

This work

3 lattice reductions, dimension \( \leq m + 1 \)

[LLKN17]

Breaking LWE

Many attractive provable security properties
THANKS!


\textbf{\textsc{NIST}} Version Attack Code: https://goo.gl/2Vo3T7
TWO-MESSAGE KEY EXCHANGE WITH STRONG SECURITY FROM IDEAL LATTICES

Yu Chen
Associate Professor
Institute of Information Engineering, Chinese Academy of Sciences
Two-message Key Exchange with Strong Security from Ideal Lattices

Zheng Yang (University of Helsinki)
Yu Chen (Chinese Academy of Sciences)
Song Luo (Chongqing University of Technology)

April 17th, CT-RSA 2018
Background: Two-message Key Exchange (TMKE)

➢ Two-message Key exchange
  • Two messages: $m_{id1}$, $m_{id2}$—derived from party’s (ephemeral) secrets.
  • Shared session key $K$—computed from party’s (ephemeral) secrets and exchanged messages
  • Appealing to practice: low bandwidth and asynchronous communication

<table>
<thead>
<tr>
<th>id₁</th>
<th>id₂</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message generation function</strong></td>
<td><strong>Protocol Execution</strong></td>
</tr>
<tr>
<td><strong>Session key generation function</strong></td>
<td>$m_{id1}$ → $m_{id2}$</td>
</tr>
<tr>
<td>accept $K$</td>
<td>← accept $K$</td>
</tr>
</tbody>
</table>
The Simplest Example of TMKE

➢ Seminal TMKE: Diffie-Hellman key exchange (DHKE) [DH76]
  • Cyclic group $G = \langle g \rangle$ of prime order $p$
  • Two messages: $X, Y$
  • Passively secure; active attacker can implement man-in-the-middle attack

\[
\begin{array}{ll}
\text{id}_1 & \text{id}_2 \\
\hline
\end{array}
\]

**Protocol Execution**

\[
x \leftarrow \mathbb{Z}_p^* \\
X := g^x \\
\]

\[
y \leftarrow \mathbb{Z}_p^* \\
Y := g^y \\
\]

accept $K := Y^x$

accept $K := X^y$
Motivation

➢ Quantum computers are about to get real
➢ DL, factoring, ...., not hard against quantum algorithms
➢ Lattice-based Cryptography
   • Quantum secure
   • Simple, efficient, and highly parallel
➢ Existing Lattice-based AKE, e.g.:
   • AsiaCCS’13, Fujioka et al.,
     — standard model, CK+ model without perfect forward secrecy (PFS)
   • Eurocrypt’15, Zhang et al.,
     — random oracle, BR model without PFS and leakage of ephemeral secret key
   • CT-RSA’14, Kurosawa and Furukawa (KF scheme)
     — standard model, eCK model without PFS  Is it Secure?
Overview of Our Results

➢ Revisit the security of the KF scheme (CT-RSA’14)
   • finding an attack

➢ Propose a new generic TMKE scheme
   • New cryptographic primitive: One-time CCA-secure KEM
   • Without random oracles
     • eCK-PFS model: known session key (KSK), key compromise impersonation (KCI), chosen identity and public key (CIDPK), ephemeral secret key leakage (ESKL), and perfect forward secrecy (PFS)

➢ Instantiation of TMKE from ideal lattices
The KF scheme

➢ Building Blocks: Twisted PRF (TPRF), Signature (SIG), IND-CPA KEM (wKEM)

\[
\begin{align*}
(id_1, id_2) & \quad (sk_{id_1}, pk_{id_1}) = ((ssk_{id_1}, s_{id_1}), spk_{id_1}) \\
\quad & \quad (sk_{id_2}, pk_{id_2}) = ((ssk_{id_2}, s_{id_2}), spk_{id_2})
\end{align*}
\]

\[\begin{array}{c}
\text{Protocol Execution} \\
\end{array}\]

\[
\begin{align*}
& r_1, r_2 \leftarrow \{0, 1\}^* \\
& R_1 \leftarrow \text{TPRF}(s_{id_1}, r_1) \\
& R_2 \leftarrow \text{TPRF}(s_{id_1}, r_2) \\
& (epk = a, epk = g^a) \leftarrow \text{wKEM.Gen}(1^\kappa, R_1) \\
& X = (id_1, epk) \\
& \sigma_X \leftarrow \text{SIG.Sign}(ssk_{id_1}, X, R_2) \\
& K := \text{wKEM.Dec}(esk_{id_1}, C)
\end{align*}
\]

\[
\begin{align*}
& C = (g^r, g^{r\cdot a} \cdot K) \leftarrow \text{wKEM.Gen}(epk_{id_1}, R_3) \\
& X, \sigma_X \quad Y, \sigma_Y X \quad \sigma_Y X \leftarrow \text{SIG.Sign}(ssk_{id_2}, Y \parallel X, R_4)
\end{align*}
\]
Insecurity of the KF scheme: Attack

\[ (sk_{id_1}, pk_{id_1}) = ((ssk_{id_1}, s_{id_1}), spk_{id_1}) \]

\( (sk_{id_2}, pk_{id_2}) = ((ssk_{id_2}, s_{id_2}), spk_{id_2}) \)

\( \pi_{id_1}^s \)

\[ R_1 \leftarrow \text{TPRF}(s_{id_1}, r_1) \]
\[ R_2 \leftarrow \text{TPRF}(s_{id_1}, r_2) \]
\( (a, g^a) \leftarrow \text{wKEM.Gen}(1^\kappa, R_1) \)
\( X = (id_1, g^a) \)
\( \sigma_X \leftarrow \text{SIG.Sign}(ssk_{id_1}, X, R_2) \)
\( \sigma_X \leftarrow \text{SIG.Sign}(ssk_{id_1}, X, R_2) \)

\( K_A := K^\beta = g^{ar\beta} \cdot K^\beta \)

\( A \) (Adversary)

\[ \pi_{id_2}^t \]

\[ R_3 \leftarrow \text{TPRF}(s_{id_1}, r_3) \]
\[ R_4 \leftarrow \text{TPRF}(s_{id_1}, r_4) \]
\( ((g^r, g^{ar} \cdot K), K) \leftarrow \text{wKEM.Gen}(g^a, R_3) \)
\( Y = (id_2, g^r, g^{ar} \cdot K) \)

\( \sigma_{Y,X} \leftarrow \text{SIG.Sign}(ssk_{id_2}, Y || X, R_4) \)

Protocol Execution

Forward \( X, \sigma_X \)
Intercept \( Y, \sigma_Y, X \)
Corrupt \( ssk_{id_2} \)
\( C_A := (g^{r\beta}, g^{ar\beta} \cdot K^\beta) \) for arbitrary \( \beta \)
\( Y_A := (id_2, C_A) \)
\( \sigma_A := \text{SIG.Sign}(ssk_{id_2}, Y_A || X, R') \)
\( X, \sigma_X \rightarrow \text{Drop} \ Y, \sigma_Y, X \)
\( X, \sigma_X \rightarrow \text{Reveal} \ K_A \) from \( \pi_{id_1} \)
\( Y, \sigma_Y, X \rightarrow \text{Compute} \ K := K_A^{\beta^{-1}} \)

\( \text{test oracle is fresh and has no partner oracle at id}_1 \)
Insecurity of the KF scheme: Problems

\( (sk_{id1}, pk_{id1}) = ((ssk_{id1}, s_{id1}), spk_{id1}) \)

\( (sk_{id2}, pk_{id2}) = ((ssk_{id2}, s_{id2}), spk_{id2}) \)

\( A \) (Adversary)

\[ \pi_{id_1}^s \]

\( R_1 \leftarrow \text{TPRF}(s_{id_1}, r_1) \)

\( R_2 \leftarrow \text{TPRF}(s_{id_1}, r_2) \)

\( (a, g^a) \leftarrow \text{wKEM.Gen}(1^\kappa, R_1) \)

\( X = (id_1, g^a) \)

\[ \sigma_X \leftarrow \text{SIG.Sign}(ssk_{id_1}, X, R_2) \]

\( \sigma_X \leftarrow \text{SIG.Sign}(ssk_{id_1}, X, R_2) \)

\( K_A := K^\beta = \frac{g^{ar^\beta} \cdot K^\beta}{g^{r^\beta}} \)

\( \pi_{id_2}^t \)

\( R_3 \leftarrow \text{TPRF}(s_{id_1}, r_3) \)

\( R_4 \leftarrow \text{TPRF}(s_{id_1}, r_4) \)

\[ ((g^r, g^{ar^r} \cdot K), K) \leftarrow \text{wKEM.Gen}(g^a, R_3) \]

\( Y = (id_2, g^r, g^{ar^r} \cdot K) \)

\( \sigma_{YX} \leftarrow \text{SIG.Sign}(ssk_{id_2}, Y || X, R_4) \)

Not tied to session info
unknown key share

One-time CCA attack against the KEM, CPA-secure KEM does not suffice
How to remedy the KF scheme?

➢ Main idea
  • Enhance the security of KEM
    • CPA to one-time CCA
  • Employ Key Derivation function
    • bind the session key with session specific information to defend active attacks
Our New Generic TMKE Protocol

➢ Building blocks

• **One-time KEM (OTKEM):** encapsulate the session key

• **Signature (SIG):** authenticate exchanged messages

• **IND-CPA KEM (wKEM):** implement NAXOS trick against ephemeral key leakage (generate the pk for OTKEM)

• **Pseudo-random function (PRF):** act as KDF to bind session key with session specific information
A New Generic TMKE Protocol

**Protocol Execution**

\[
\begin{align*}
&c_{id_1} \leftarrow \mathcal{C}_{wKEM}, \quad r_{s_{id_1}} \leftarrow \mathcal{R}_{SIG} \\
r_{pg_{id_1}} \leftarrow wKEM.Dec(d_{k_{id_1}}, c_{id_1}) \\
(e_{pk_{id_1}}, e_{sk_{id_1}}) \leftarrow OTKEM.Gen(1^r, r_{pg_{id_1}}) \\
\sigma_{id_1} \leftarrow SIG.Sign(s_{sk_{id_1}}, e_{pk_{id_1}}, r_{s_{id_1}}) \\
m_{id_1} := (id_1, e_{pk_{id_1}}, \sigma_{id_1})
\end{align*}
\]

\[
\begin{align*}
&c_{id_2} \leftarrow \mathcal{C}_{wKEM}, \quad r_{s_{id_2}} \leftarrow \mathcal{R}_{SIG} \\
er_{k_{id_2}} \leftarrow wKEM.Dec(d_{k_{id_2}}, c_{id_2}) \\
(k, C_{id_2}) \leftarrow OTKEM.Enc(e_{pk_{id_1}}, er_{k_{id_2}}) \\
T := id_1||p_{k_{id_1}}||e_{pk_{id_1}}||\sigma_{id_1}||id_2||p_{k_{id_2}}||C_{id_2} \\
\sigma_{id_2} \leftarrow SIG.Sign(s_{sk_{id_2}}, T, r_{s_{id_2}}) \\
m_{id_2} := (id_2, C_{id_2}, \sigma_{id_2})
\end{align*}
\]

reject if SIG.Vfy(s_{pk_{id_1}}, \sigma_{id_1}, e_{pk_{id_1}}) \neq 1

\[
\begin{align*}
&k \leftarrow OTKEM.Dec(e_{sk_{id_1}}, C_{id_2}) \\
\text{sid} := T||\sigma_{id_2} \\
&\text{accept } K := \text{PRF}(k, \text{sid})
\end{align*}
\]

\[
\begin{align*}
&\text{reject if SIG.Vfy(s_{pk_{id_2}}, \sigma_{id_2}, T) \neq 1} \\
&k \leftarrow OTKEM.Dec(e_{sk_{id_1}}, C_{id_2}) \\
\text{sid} := T||\sigma_{id_2} \\
&\text{accept } K := \text{PRF}(k, \text{sid})
\end{align*}
\]
Instantiations from Ideal Lattices

- Building blocks’ instantiations from existing works:
  - **Signature (SIG):** Ruckert (PQCrypto’10)
  - **IND-CPA KEM (wKEM):** Peikert (PQCrypto’14).
  - **Pseudo-random function (PRF):** Banrjee et al. (Eurocrypt’12)
  - **One-time KEM (OTKEM):** q-bounded IND-CCA KEM (q=1), Cramer et al. Asiacrypt’07 (less efficient)

Can we build efficient OTKEM from ideal lattices?
Efficient OTKEM from Ideal Lattices

- Direct construction
  - Ring-Learning with Errors (RLWE):
    \[ a \in \mathcal{R}, (s, e) \in \mathcal{X}, V_0 := a \cdot s + e, V_1 \in \mathcal{R} \]
  - Target collision resistant hash function (TCRHF):
    \[ \text{TCRHF} : hk_{\text{TCRHF}} \times R_q \rightarrow \{0, 1\}^n \]

Similar to construction of OTS from OWF
Thank you very much for your attention!