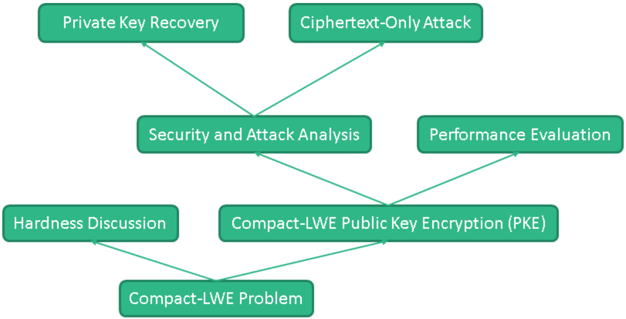
Compact-LWE: Lattice-based PKE without Concretely Relying on the Hardness of Lattice Problems

Dongxi Liu   Nan Li   Jongkil Kim   Surya Nepal

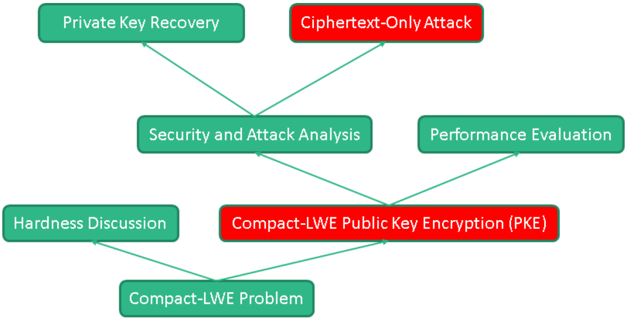
9 April 2018

Contents in Submitted Specification



Security Problem

- Ciphertext-only attacks to Compact-LWE PKE can be true
  - Found by Pan et al., and Boole et al.
  - A countermeasure provided below
- Hardness of Compact-LWE problem not affected



Outline

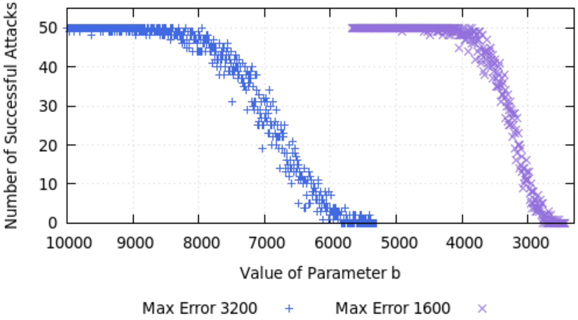
- Compact-LWE problem and its hardness
- Compact-LWE PKE
  - key generation, encryption, decryption
  - an instance (parameters, sizes of keys and ciphertexts)
- Explanation of Ciphertext-only Attack
- Countermeasure: Revision to Compact-LWE PKE
- Advantages

Compact-LWE problem

- Secret values: $\mathbf{s}, \mathbf{s}' \in \mathbb{Z}_q^n$, $k, k' \in \mathbb{Z}_q$, $ck, ck' \in \mathbb{Z}_p$, and $p < q$
  - All values randomly sampled from uniform distributions
- Compact-LWE samples
  - $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}\rangle + k*(r_i + p*e_i) \bmod q, \langle \mathbf{a}_i, \mathbf{s}'\rangle + k'*(r_i' + p*e_i') \bmod q)$
    - $r_i, r_i' \in \mathbb{Z}_p$, satisfying $ck * r_i + ck' * r_i' = 0 \texttt{ mod } p$
    - e_i, e_i' are small error values
    - $\mathbf{a}_i \in \mathbb{Z}_b^n$, with $b \leq q$
- Compact-LWE problem
  - finding secret values from Compact-LWE samples

Hardness of Compact-LWE Problem

- A LWE sample is a Compact-LWE sample with $k = 1$, $k' = 1$, $p = 1$, $ck = 0$, $ck' = 0$, and $b = q$.
- Smaller b makes Compact-LWE resistant to lattice-based attacks to recover original $\mathbf{s}$ or $\mathbf{s}'$.



Compact-LWE PKE

- Public parameters: nine positive integers
  - $q, t, n, m, w, w', b, b', l$
- Generation of private keys
  - Private parameters: sk_max, p_size, e_min, and e_max
  - Private key: $(\mathbf{s}, k, sk, ck, \mathbf{s}', k', sk', ck', p)$
    - $p \in \{(w + w') * b', ..., (w + w') * b' + p\_size\}$
    - p coprime with q and $sk\_max * b' + p + e\_max * p < q/(w + w')$
    - $sk, sk' \in \mathbb{Z}_{sk\_max}$, satisfying $sk * ck + sk' * ck'$ coprime with p

Compact-LWE PKE

- Generation of public keys
  - m public key samples $(\mathbf{a}_i, u_i, pk_i, pk'_i)$
    - $pk_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + k_q^{-1} * (sk * u_i + r_i + e_i * p) \bmod q$
    - $pk'_i = \langle \mathbf{a}_i, \mathbf{s}' \rangle + {k'}_q^{-1} * (sk' * u_i + r'_i + e'_i * p) \bmod q$
    - $u_i \in \mathbb{Z}_{b'}$, $e_i \in [e\_min, e\_max]$, and $e'_i \in [e\_min, e\_max]$
- Encryption
  - basic encryption: only encrypting messages in $\mathbb{Z}_t$
  - general encryption: relying on basic encryption to encrypt long messages

Compact-LWE PKE: Basic Encryption

- Generate the m-dimensional random vector $\mathbf{l}$, such that
  - $w \leq \Sigma_{i=1}^{m} \mathbf{l}[i] \leq w + w'$ for all $\mathbf{l}[i] > 0$
  - $-w' \leq \Sigma_{i=1}^{m} \mathbf{l}[i] \leq 0$ for all $\mathbf{l}[i] < 0$
  - $\Sigma_{i=1}^{m} \mathbf{l}[i] * u_i > 0$
- Generate the ciphertext $\mathbf{c}$
  - $(\Sigma_{i=1}^{m} \mathbf{l}[i] * \mathbf{a}_i, f(v, \Sigma_{i=1}^{m} \mathbf{l}[i] * u_i), \Sigma_{i=1}^{m} \mathbf{l}[i] * pk_i \bmod q,$ $\Sigma_{i=1}^{m} \mathbf{l}[i] * pk_i' \bmod q)$
    - where

$$f(v, \Sigma_{i=1}^{m} \mathbf{l}[i] * u_i) = (v \oplus \mathtt{rol}(u, \log_2(t)/2)) * u' \bmod t,$$

$$u = (\Sigma_{i=1}^{m} \mathbf{l}[i] * u_i) \bmod t, \text{ and}$$

$$u' \geq (\Sigma_{i=1}^{m} \mathbf{l}[i] * u_i)/t \text{ is the smallest integer coprime with t.}$$

Compact-LWE PKE: Basic Decryption

- Let $\mathbf{c} = (\mathbf{a}, d, pk, pk')$ be the ciphertext.
- With the private key, v is recovered by using the steps below:
  - Calculate $d_1 = (pk - \langle \mathbf{a}, \mathbf{s} \rangle) * k \bmod q$, and $d_1' = (pk' - \langle \mathbf{a}, \mathbf{s}' \rangle) * k' \bmod q$.
  - Let $d_2 = ck * d_1 + ck' * d_1' \bmod p$.
  - Calculate $d_3 = sckInv * d_2 \bmod p$, where $sckInv$ is determined by $sckInv * (sk * ck + sk' * ck') = 1 \bmod p$.
  - Obtain $v = f^{-1}(d, d_3)$, where

$$f^{-1}(d, d_3) = ({u'}_p^{-1} * d \bmod t) \oplus \mathtt{rol}(u, \log_2(t)/2),$$

$$u = d_3 \bmod t,$$

$$u' \geq d_3/t \text{ is the smallest integer coprime with t, and}$$

$$ {u'}_p^{-1} * u' = 1 \bmod t.$$

An Instance: parameters

- 192-bit search space for private keys

q	t	n	m	w	w'	b	b'	l
2^{64}	2^{32}	8	128	224	32	16	68719476736	8

Table: Public Parameters

sk_max	p_size	e_min	e_max
229119	16777216	457	3200

Table: Private Parameters

An Instance: sizes and performance

- 232 bytes for a private keys and 2064 bytes for a public key

Message (B)	32	64	128	256	512	1024
Ciphertext (B)	360	648	1224	2376	4680	9288

Table: Ciphertext Size

Message (B)	32	64	128	256	512	1024
Enc (sec)	1.29	2.15	4.36	7.56	14.81	28.7
Dec (sec)	0.18	0.27	0.43	0.88	1.78	3.50

Table: Performance of 10000 Encryptions and Decryptions

- Note that the evaluation will change in the revised version of Compact-LWE encryption scheme.

Explanation of Ciphertext-only Attack

- Given ciphertext $\mathbf{c} = (\mathbf{a}, d, pk, pk')$, we have
  - $\mathbf{a} = \Sigma_{i=1}^{m} \mathbf{l}[i] * \mathbf{a}_i$
  - $pk = \Sigma_{i=1}^{m} \mathbf{l}[i] * pk_i \mod q$
  - $pk' = \Sigma_{i=1}^{m} \mathbf{l}[i] * pk'_i \mod q$
  - $d = f(v, \Sigma_{i=1}^{m} \mathbf{l}[i] * u_i)$
- From the first three equations, a short vector $\mathbf{l}'$ can be obtained.
- The ciphetrext-only attack can succeed, due to $\Sigma_{i=1}^{m} \mathbf{l}[i] * u_i = \Sigma_{i=1}^{m} \mathbf{l}'[i] * u_i$.

Compact-LWE PKE - revised

- Changes indicated in red.
- Public parameters: ten positive integers
  - $q, t, n, m, w, w', b, b', l, n'$
- Generation of private keys
  - Private parameters: $\cancel{sk\_max}$, p_size, $\cancel{e\_min}$, and e_max
  - Private key: $(\mathbf{s}, k, sk, ck, \mathbf{s}', k', sk', ck', p, \mathbf{s}'')$
    - $p \in \{(w + w') * b', ..., (w + w') * b' + p\_size\}$
    - p coprime with q and $p + p + e\_max * p < q/(w + w')$
    - $sk, sk' \in \mathbb{Z}_p$, satisfying $sk * ck = sk' * ck'$ and $sk * ck$ coprime with p
    - $\mathbf{s}'' = (\mathbf{s}''[1], ..., \mathbf{s}''[n']) \in \mathbb{Z}_{b'}^{n'}$, with $\mathbf{s}''[1]$ and $\mathbf{s}''[2]$ co-prime with b'

Compact-LWE PKE - revised

- Generation of public keys
  - m public key samples $(\mathbf{a}_i, \mathbf{a'}_i, pk_i, pk'_i)$
    - $pk_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + k_q^{-1} * ((sk * u_i \bmod p) + r_i + e_i * p) \bmod q$
    - $pk'_i = \langle \mathbf{a}_i, \mathbf{s}' \rangle + {k'}_q^{-1} * ((sk' * u'_i \bmod p) + r'_i + e'_i * p) \bmod q$
    - $u_i, u'_i \in \mathbb{Z}_p$, and $(u_i + u'_i) \bmod p \in \mathbb{Z}_{b'}$
    - $e_i \in [0, e\_max]$, and $e'_i \in [0, e\_max]$
    - $\mathbf{a'}_i \in \mathbb{Z}_{b'}^{n'}$, and $\langle \mathbf{a'}_i, \mathbf{s''} \rangle = ((u_i + u'_i) \bmod p) \bmod b'$
  - Let $\mathbf{s2''} = (\mathbf{s''}[1] * \mathbf{s''}[1], ..., \mathbf{s''}[n'] * \mathbf{s''}[n']) \in \mathbb{Z}_{b'}^{n'}$.
  - For $1 \leq i < j \leq n'$, $\mathbf{a''}_{ij} \in \mathbb{Z}_{b'}^{n'}$ are included in the public key,
    - Satisfying $\langle \mathbf{a''}_{ij}, \mathbf{s2''} \rangle = \mathbf{s''}[i] * \mathbf{s''}[j] \bmod \text{b'}$
- Encryption
  - basic encryption: only encrypting messages in $\mathbb{Z}_t$
  - general encryption: relying on basic encryption to encrypt long messages

Compact-LWE PKE: Basic Encryption - revised

- Generate the m-dimensional random vector $\mathbf{l}$, such that
  - $w \leq \Sigma_{i=1}^{m} \mathbf{l}[i] \leq w + w'$ and $\mathbf{l}[i] > 0$ for $1 \leq i \leq m$
  - $\cancel{-w' \leq \Sigma_{i=1}^{m} \mathbf{l}[i] \leq 0 \text{ for all } \mathbf{l}[i] < 0}$
  - $\cancel{\Sigma_{i=1}^{m} \mathbf{l}[i] * u_i > 0}$
- Generate the ciphertext $\mathbf{c}$
  - $(\Sigma_{i=1}^{m} \mathbf{l}[i] * \mathbf{a}_i, \mathbf{a}', f(v, u), \Sigma_{i=1}^{m} \mathbf{l}[i] * pk_i \bmod q, \Sigma_{i=1}^{m} \mathbf{l}[i] * pk'_i \bmod q)$, *where*
    - u randomly sampled from $\mathbb{Z}_{b'}$
    - let $(a_1, \ldots, a_{n'}) = \Sigma_{i=1}^{m} \mathbf{l}[i] * \mathbf{a}'_i$
    - $\mathbf{a}' = (a_1^2, \ldots, a_{n'}^2) + \Sigma_{i=1}^{n'-1} \Sigma_{j=i+1}^{n'} 2 * a_i * a_j * \mathbf{a}''_{ij} + u * \mathbf{a}''_{12} \in \mathbb{Z}_{b'}^{n'}$
    - no change to f
- More random u (e.g., $u' * \mathbf{a}_{13} + u'' * \mathbf{a}_{14} + \ldots$) can be added into $\mathbf{a}'$ if general encryption and decryption are also revised.

Compact-LWE PKE: Basic Decryption - revised

- Let $\mathbf{c} = (\mathbf{a}, \mathbf{a}', d, pk, pk')$ be the ciphertext.
- With the private key, v is recovered by using the steps below:
  - Calculate $d_1 = (pk - \langle \mathbf{a}, \mathbf{s} \rangle) * k \bmod q$, and $d_1' = (pk' - \langle \mathbf{a}, \mathbf{s}' \rangle) * k' \bmod q$.
  - Let $d_2 = ck * d_1 + ck' * d_1' \bmod p$.
  - Calculate $d_3 = sckInv * d_2 \bmod p$, where $sckInv$ is determined by $sckInv * (sk * ck + sk' * ck') = 1 \bmod p$.
  - Let $\mathbf{s2}'' = (\mathbf{s}''[1] * \mathbf{s}''[1], \ldots, \mathbf{s}''[n'] * \mathbf{s}''[n']) \in \mathbb{Z}_{b'}^{n'}$.
  - Calculate $u = (\mathbf{s}''[1] * \mathbf{s}''[2])^{-1} * (\langle \mathbf{a}', \mathbf{s2}'' \rangle - d_3 * d_3) \bmod b'$
  - Obtain $v = f^{-1}(d, u)$

Evaluation of Countermeasure

- Implementation of basic encryption and decryption in Sage.
- $(n' - 1) * \log_2 b'$ should be greater than the declared security level.
  - $n' = 6$ and $b' = 2^{39}$ used in our evaluation $(5 * 39 > 192)$
- $n' - 1$ elements in $\mathbf{a}'_i$ are independently and randomly sampled.
  - The idea of ciphertext-only attack explained before is not applicable.
    - i.e., $\Sigma_{i=1}^{m} \mathbf{l}[i] * \mathbf{a}'_i \neq \Sigma_{i=1}^{m} \mathbf{l}'[i] * \mathbf{a}'_i$ when $n' > 1$.
- b' must be a composite number.
  - $\mathbb{Z}_{b'}$ is not a field, and thus $\mathbf{s}''$ cannot be recovered from $\mathbf{a}''_{ij}$ by solving MQ equations.
  - $\mathbf{a}''_{ij}$ is used to reduce ciphertext size; i.e., without $\mathbf{a}''_{ij}$, ciphertexts become bigger.

Advantages

- Simple to understand and implement
  - Constructed with integers and modular arithmetic
  - All random values sampled from uniform distribution
- Assuming hard problems in lattices can be efficiently solved, with small parameters (e.g., n=8) selected
  - Detect design flaws if there are easily with concrete attacks (good for preventing deeply hidden design flaws)
  - Mitigate the impact when hard problems in lattices become not hard in future
- Relatively small ciphertexts
  - A ciphertext has about 700 bytes for a 32-byte message in the revised version.