6.892 Computing on Encrypted Data September 16, 2013

Lecture 2

Lecturer: Vinod Vaikuntanathan *Scribe: Britt Cyr*

In this lecture, we will define the learning with errors (LWE) problem, show an equivalence between the search and decision versions of LWE, construct encryption schemes based on the hardness of LWE, and begin to see their homomorphic properties. We start with definitions.

1 Definitions

Definition 1 (Search $LWE_{n,q,\chi}$)**.**

Search LWE is the Learning with Errors problem with the goal to find $\vec{s}$
n is the security parameter
q is an odd modulus
χ is the probability distribution on $\mathbb{Z}_q = \{-\frac{q-1}{2}, \ldots \frac{q-1}{2}\}$

Definition 2 (B-Bounded)**.**

We will say that χ is B-bounded if $Pr_{x \leftarrow \chi}[|x| > B] = negl(n)$

Definition 3 (Search LWE assumption)**.**

Given access to oracle, it must be computationally hard to find s.

$$\forall PPT A, Pr_s[A^{LWE_s}(1^n) = s] = negl(n)$$

Where the input to LWE_s is

$$(a \leftarrow \mathbb{Z}_q^n, \langle a, s \rangle + e) \text{ where } e \leftarrow \chi$$

Definition 4 (Decision LWE assumption)**.**

Cannot distinguish samples of LWE and from uniform random

$$\left| Pr[D^{LWE_s}(1^n) = 1] - Pr[D^{random}(1^n) = 1] \right| = negl(n)$$

1.1 LWE Reductions

The reduction from search to decision is trivial.
Decision is as hard as search (proven later)
Search LWE (average case) $\rightarrow$ approximate gaps V_p on any lattice (worst case)
Search LWE (average case) $\rightarrow$ shortest independent vectors problem (worst case)

2 Encryption with LWE

2.1 OWF based on hardness of LWE

$$f(A, s, e) = (A, As + e)$$
$$A \in \mathbb{Z}_q^{m \times n}$$
$$e \in \chi^m$$

This is an injective OWF for $m >> n$, where m is the number of samples, since it is hard to find s by the search LWE assumption.

2.2 PRG

Decision LWE gives a PRG

$$f : (A, s, e) \to (A, As + e)$$

Consider the number of bits, ignore A since it is on both sides

$$n \log q + m \log B \to m \log q$$

This is a PRG for $m >> n$ since then $n \log q + m \log B < m \log q$

2.3 Private Key Encryption

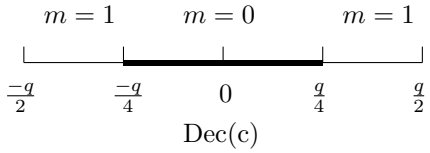
Sending a single bit

$$SK = \vec{s} \leftarrow \mathbb{Z}_q^n$$

$$Enc(SK, m) = (a, \langle a, s \rangle + e + m(\frac{q+1}{2}) \mod q) \text{ with a and e chosen at random}$$

$$Dec(c) = b - \langle a, s \rangle = e + m(\frac{q+1}{2})$$

Decrypts to 0 if the result is small and 1 if it is near $\frac{q}{2}$. This works if $|e| < \frac{q}{4}$.



2.4 Public Key Encryption

In general, we can begin with a secret key homomorphic encryption scheme and create a public key homomorphic encryption scheme from it. This does not hold if the encryption scheme is not homomorphic.

2.4.1 First attempt

For the public key, publish an encryption of 0 and an encryption of 1. Then the person doing the encrypting can choose which one to use.

This however does not work because the same key is used every time and the adversary can detect a repeat.

2.4.2 Modification

This can be modified to be correct by instead publishing many encryptions of 0. The idea is to take a random linear combination of these to get a fresh encryption of 0 for each use. [Regev '05]

$$SK = \vec{s} \leftarrow \mathbb{Z}_q^n$$

$$PK = (A, As + e) \text{ where } A \leftarrow \mathbb{Z}_q^{m \times n}$$

$$Enc(\mu) = (rA, r(As + e) + \mu(\frac{q+1}{2}) \mod q)$$

$$Dec(c) = b - \langle a, s \rangle = e + \mu(\frac{q+1}{2})$$

Encryption takes $r \leftarrow \{0,1\}^m$ and makes a random linear combination of the public key to get a new encryption of 0. Decryption is the same as the private key case. If the output is near zero, it is a 0, if it is near $\frac{q}{2}$, then it is a 1.

This will correctly decrypt as long as $|\langle r, e\rangle| < \frac{q}{4}$, when the error is not too large. This is true if $|e_i| < \frac{q}{4m}$.

2.5 Security Argument

Because of the leftover hash lemma [ILL '89]

$$(PK, Enc_{PK}(0)) \approx_c (\tilde{PK}, Enc_{\tilde{PK}}(0))$$

$\tilde{PK} = (A, y)$ where $A \leftarrow \mathbb{Z}_q^{m \times n}$ and $y \leftarrow \mathbb{Z}_q^m$
By the Decision LWE Assumption

$$(\tilde{PK}, Enc_{\tilde{PK}}(0)) \approx_s \text{ Uniform Random}$$

Similarly, we can show

$$(PK, Enc_{PK}(1)) \approx_c (\tilde{PK}, Enc_{\tilde{PK}}(1))$$

$$(\tilde{PK}, Enc_{\tilde{PK}}(1)) \approx_s \text{ Uniform Random}$$

Therefore the encryption is semantically secure

$$(PK, Enc_{PK}(0)) \approx_c (PK, Enc_{PK}(1))$$

2.6 Leftover Hash Lemma [ILL '89]

Given

$$A \leftarrow \mathbb{Z}_q^{m \times (n+1)}$$

$$r \leftarrow \{0,1\}^m$$

$$b \leftarrow \mathbb{Z}_q^{n+1} \text{ uniformly random}$$

Then

$$(A, rA) \approx_\epsilon (A, b)$$

2.7 Additive Homomorphism

$$c_1 = (a_1, \langle a_1, s\rangle + e_1 + m_1(\frac{q+1}{2}))$$

$$c_2 = (a_2, \langle a_2, s\rangle + e_2 + m_2(\frac{q+1}{2}))$$

To add, just add the two ciphers. Ignore the $m_1 m_2$ term because it is negligible since we are just adding single bits and q is big

$$c_1 + c_2 = (a_1 + a_2, \langle a_1 + a_2, s\rangle + e_1 + e_2 + (m_1 \oplus m_2)\frac{q+1}{2})$$

We need to set appropriately sized parameters so errors do not get too large
q is subexponential
e is small polynomial
We can add A different ciphertexts as long as $B = O(\frac{q}{A})$

3 Detour: Search to Decision Reduction for LWE

Goal:

$$LWE_{n,q,\chi,m} \leq dLWE_{n,q,\chi,m'}$$
$$m = poly(n, m', \frac{1}{\epsilon})$$
$$t = poly(n, q, m')$$

3.1 Reduction [Blum, Furst, Kearns, Lipton '93]

Good	Bad
Any Error Distribution	q must be prime
No advantage for quantum algorithms	Time is poly in q, not $\log q$ [Brakerski '13]
	m grows with $\frac{1}{\epsilon}$ [Micciancio, Mol '11]

3.2 Overview of Reduction

The reduction is in 3 steps. Beginning with a machine for dLWE for a random $\vec{s}$, create a machine that works for any $\vec{s}$. Then amplify the advantage to make it a nearly perfect decider. Then with that machine, make a decider that solves search with advantage $\frac{1}{2}$.

D_0 :$dLWE_{n,q,x,m'}$ advantage ϵ for a random s
D_1 :$dLWE_{n,q,x,m'}$ advantage ϵ for any s
D_2 :$dLWE_{n,q,x,m}$ advantage $1 - O(\frac{n}{q})$
D_3 :$searchLWE_{n,q,\chi,m}$ advantage $\frac{1}{2}$

3.3 First Reduction: $D_0 \to D_1$

Pick from uniform random $t \leftarrow \mathbb{Z}_q^n$.

Let $b = As + e$. Give the machine $(A, b + At)$. This confuses the machine enough to make it a decider for any s.

3.4 Second Reduction: $D_1 \to D_2$

Split m' into chunks of m and run each which is the number of samples used in each instance of D_1

Run D_1 for $O(\frac{\log nq}{\epsilon})$ iterations with these independent sets of samples. Each result casts a vote with probability of being correct $\frac{1}{2} + \epsilon$. Using a Chernoff bound, the probability that the result of all the votes is correct is $1 - O(\frac{1}{nq})$. This is amplification in a braindead way.

3.5 Third Reduction: $D_2 \rightarrow D_3$

The method for this reduction from decision to search will be guess and check. For every coordinate s_i, step through all possible values.

for $i = 1$ **to** n **do**
  for $g_i = 1$ **to** $q - 1$ **do**
    sample random $r \leftarrow Z_q$
    Feed D_2 with $(a' = a + r \cdot u_i, b' = \langle a, s \rangle + e + rg_i)$ where u_i is zero vector except 1 in i^{th} location
    if D_2 says random **then**
      step through
    else
      set $s_i = g_i$
    end if
  end for
end for

3.6 Correctness of Reduction

Claim 1 (If $g_i = s_i$, then set of samples $\approx$ LWE.)**.**

The correct LWE is $(a', b') = (a', (\langle a, s \rangle + e) + r \cdot s_i)$. So if $s_i = g_i$, then they are the same.

Claim 2 (If $g_1 \neq s_i$ then the set of samples $\approx$ random and so D_2 will identify as random.)**.**

$$(a', b' = \langle a', s \rangle + e + r \cdot (g_i - s_i))$$

r is taken at random from $\mathbb{Z}_q$ and $g_i - s_i$ is non-zero in this case. Since we are working in a prime field, this whole term is random. This random term can be as big as q since r can be. Therefore b' is indistinguishable from random.

3.7 Best LWE algorithm

Lattice-basis reduction techniques [Peikert, Micciancio '13]
Parameters

$$sLWE_{n,q,\chi,poly(m)}$$

$$t = 2^{\tilde{O}^{\frac{n}{\log \frac{q}{B}}}}$$

When we set B and q as

$$B = poly(n)$$

$$q = 2^{O(n^{\epsilon})}$$

$$t = 2^{\tilde{O}(n^{1-\epsilon})}$$

We can set q as large as $2^{O(\sqrt{n})}$ for decision LWE

4 Multiplicative homomorphism

There are two ways of looking at ciphertexts the encryption view and the decryption view.

4.1 Encryption View

This view thinks of the ciphertext as the encryption of a message

$$c_1 = (a_1, \langle a_1, s\rangle + e_1 + m_1 \frac{q+1}{2})$$

$$c_2 = (a_2, \langle a_2, s\rangle + e_2 + m_2 \frac{q+1}{2})$$

4.2 Decrytion View

This view thinks of the ciphertext as a preimage of the decryption function

$$\langle c_1, t\rangle = e_1 + m_1 \frac{q+1}{2}$$

$$\langle c_2, t\rangle = e_2 + m_2 \frac{q+1}{2}$$

Where $|e_1|$ and $|e_2|$ are both small

4.3 Homomorphism

Notation:

$$\begin{aligned} t &= (-s, 1) \in \mathbb{Z}_q^{n+1} \\ &= \langle 2c_1 \otimes c_2, t \otimes t\rangle \end{aligned}$$

Where $\otimes$ is the tensor product and this expands to a quadratic polynomial of the original size

It is better to consider the decryption view for the multiplicative homomorphism. The product of the ciphertexts does not need to look like the original ciphertexts.

$$\begin{aligned} 2\cdot < c_1, t > \cdot < c_2, t > &= (2e_1 + m_1)(e_2 + m_2 \frac{q+1}{2}) \\ &= (2e_1e_2 + m_1e_2 + e_1m_2) + m_1m_2(\frac{q+1}{2}) \end{aligned}$$

Since the messages are just bits, and the product of the errors is small, we can ignore all but $m_1m_2(\frac{q+1}{2})$ which is the product of the two messages.