CS 355 Lecture 11 (5/7)

Logistics: HW3 due Friday (5/11)
HW4 posted this Friday

[Also: some changes to Sam's and Henry's office hours — see Piazza]

Previous lectures: SIS $\Rightarrow$ OWFs, CRHFs (symmetric cryptography)
LWE $\Rightarrow$ PKE, key exchange (public-key cryptography) [HW4]

} But we know how to do all of this before from number-theory (e.g., DDH, RSA, etc.)

Question: Do lattices give us additional power that we did not have before?

This lecture: Fully homomorphic encryption (FHE)

"Can we compute on encrypted data" — very useful for outsourcing computation (e.g. "encrypted Google search")

Abstractly: given encryption ctx of value x under some public key, can we derive from that an encryption of $f(x)$ for an arbitrary function f?

Ex. ElGamal encryption: (message in exponent) pk: $(g, h = g^s)$ sk: s

$\text{Enc}(pk, x_1): (g^{r_1}, h^{r_1} g^{x_1})$
$\text{Enc}(pk, x_2): (g^{r_2}, h^{r_2} g^{x_2})$
$\Rightarrow \underbrace{(g^{r_1+r_2}, h^{r_1+r_2} g^{x_1+x_2})}_{\text{encryption of sum } x_1+x_2}$ [ElGamal is additively homomorphic]

Ex. Regev encryption: pk: $(A, b^T = s^T A + e^T)$ sk: s

$\text{Enc}(pk, x_1): (Ar_1, b^T r_1 + x_1 \cdot \lfloor \frac{q}{2} \rceil)$
$\text{Enc}(pk, x_2): (Ar_2, b^T r_2 + x_2 \cdot \lfloor \frac{q}{2} \rceil)$
$\Rightarrow \underbrace{(A(r_1+r_2), b^T(r_1+r_2) + (x_1+x_2) \cdot \lfloor \frac{q}{2} \rceil)}_{\text{encryption of sum } x_1 + x_2 \in \mathbb{Z}_2}$

Note: in this lecture, we will write the LWE assumption as

$$(A, s^T A + e^T) \overset{c}{\approx} (A, u)$$

$A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $s \xleftarrow{R} \mathbb{Z}_q^n$, $u \xleftarrow{R} \mathbb{Z}_q^m$, $e \leftarrow \chi_B$ — this is the same assumption as in previous lectures, just transposed (oftentimes, this is a more convenient form for modern lattice constructions)

In both of these cases, we can evaluate single operation on ciphertexts (e.g., addition or multiplication) Can we support both addition and multiplication?

$\Rightarrow$ Fully homomorphic encryption: encryption scheme that supports both addition and multiplication on ciphertexts (thus, suffices for arbitrary computation)

Major open problem in cryptography (dates back to late 1970s!) — first solved by Stanford student Craig Gentry in 2009

↳ revolutionized lattice-based cryptography!

General blueprint: 1. Build somewhat homomorphic encryption (SWHE) — encryption scheme that supports bounded number of homomorphic operations

2. Bootstrap SWHE to FHE (essentially a way to "refresh" ciphertext)

Focus will be on building SWHE (has all of the ingredients for realizing FHE)

↳ In particular, will present Gentry-Sahai-Waters (GSW) construction (conceptually simplest scheme, though not the most concretely efficient)
"3rd generation of FHE"

Starting point: Regev encryption

KeyGen(1^λ): $\tilde{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ $\qquad A = \begin{bmatrix} \tilde{A} \\ \tilde{s}^T \tilde{A} + e^T \end{bmatrix} \in \mathbb{Z}_q^{(n+1) \times m}$

$\tilde{s} \xleftarrow{R} \mathbb{Z}_q^n$

$e \xleftarrow{R} \chi^m \qquad s = \begin{bmatrix} -\tilde{s} \\ 1 \end{bmatrix} \in \mathbb{Z}_q^{n+1}$

output $pk = A$ and $sk = s$

Observation:

$$s^T A = -\tilde{s}^T \tilde{A} + \tilde{s}^T \tilde{A} + e^T = e^T \approx 0^m$$

Encrypt(pk, x): Write $pk = A \in \mathbb{Z}_q^{(n+1) \times m}$ and sample $R \xleftarrow{R} \{0,1\}^{m \times m}$

$$C = AR + x \cdot \lfloor \tfrac{q}{2} \rceil \cdot I_{(n+1) \times m}$$

$$I_{(n+1) \times m} = \big(\underbrace{\begin{smallmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{smallmatrix}}_{\substack{(n+1) \times (n+1) \\ \text{identity matrix}}} \big| \, 0 \big)$$

Decrypt(sk, c): Write $sk = s$. Compute $s^T C$ and output 0 if $|(s^T C)_{n+1}| < \frac{q}{4}$ and 1 if $|(s^T C)_{n+1}| > \frac{q}{4}$

$(n+1)^{st}$ component of $s^T C$, interpreted as value between $-\frac{q}{2}$ and $\frac{q}{2}$

Correctness:
$$\begin{aligned} s^T C &= s^T A R + x \cdot \lfloor \tfrac{q}{2} \rceil \cdot s^T I_{(n+1) \times m} \\ &= e^T R + x \cdot \lfloor \tfrac{q}{2} \rceil \cdot s^T \\ &\approx x \cdot \lfloor \tfrac{q}{2} \rceil \cdot s^T \end{aligned}$$

Observe: the vector s (i.e., the secret key) is an approximate left-eigenvector of the matrix C (i.e., the ciphertext) with associated eigenvalue $x \cdot \lfloor \frac{q}{2} \rceil$ (i.e., the "encoded" message)

Security: Same as proof for Regev encryption (two hybrids: LWE, then LHL)

Observe: We can pad A with rows of all-zeroes so it is a square matrix (over $\mathbb{Z}_q^{m \times m}$) and pad s accordingly as well

For the ciphertext, we just embed the message in the first $(n+1)$ components

Then, correctness and security follow as before (scheme has not changed), and the message is simply the "noisy" eigenvalue associated with s (the "noisy" eigenvector)

Why is this view useful? Because eigenvalues add and multiply:

- Suppose x_1 is a (left) eigenvalue of C_1 with associated eigenvector s
- Suppose x_2 is a (left) eigenvalue of C_2 with associated eigenvector s

Then: $s^T (C_1 + C_2) = s^T C_1 + s^T C_2 = x_1 s^T + x_2 s^T = (x_1 + x_2) s^T$

$s^T C_1 C_2 = x_1 \cdot s^T C_2 = x_1 x_2 s^T$

Enables homomorphic operations!

Does the above work with approximate eigenvalues (with the padded matrices)? Unfortunately, not... Need new tricks!

Correctness: $s^T C = x \cdot \lfloor \frac{q}{2} \rceil \cdot s^T + e^T R$

Addition: $s^T (C_1 + C_2) = x_1 \cdot \lfloor \frac{q}{2} \rceil \cdot s^T + e^T R_1 + x_2 \cdot \lfloor \frac{q}{2} \rceil \cdot s^T + e^T R_2$

$= (x_1 + x_2) \cdot \lfloor \frac{q}{2} \rceil \cdot s^T + e^T (R_1 + R_2)$ Works as long as $R_1 + R_2$ is small! (As long as $B \ll q$, this will be OK.)

Multiplication: $s^T C_1 C_2 = (x_1 \cdot \lfloor \frac{q}{2} \rceil \cdot s^T + e^T R_1) C_2 = x_1 \cdot \lfloor \frac{q}{2} \rceil \cdot s^T C_2 + e^T R_1 C_2$

$$= \underbrace{x_1 \cdot \lfloor \tfrac{q}{2} \rceil \cdot (x_2 \cdot \lfloor \tfrac{q}{2} \rceil + e^T R_2)}_{\substack{\text{not quite what we wanted due} \\ \text{to the message encoding, but} \\ \text{should be fixable...}}} + \underbrace{e^T R_1 C_2}_{\text{this is large since } C_2 \text{ is not short!}}$$

$\hookrightarrow$ Correctness fails for multiplication!

The gadget matrix: A matrix with a public trapdoor (can also be viewed as a "powers-of-two" matrix)

$$G = \begin{pmatrix} 1\ 2\ 4 \cdots 2^{\lfloor \log q \rfloor} & & 0 \\ & 1\ 2\ 4 \cdots 2^{\lfloor \log q \rfloor} & \\ 0 & & \ddots \end{pmatrix} \in \mathbb{Z}_q^{n \times n \lfloor \log q \rfloor}$$

For notational simplicity, we will write $m = n\lfloor \log q \rfloor = \Theta(n \log q)$

G is a matrix with the powers-of-two along the diagonal

a more compact way to write this is $G = (1\ 2\ 4 \cdots 2^{\lfloor \log q \rfloor}) \otimes I_n$

The magic of the gadget matrix: given any $v \in \mathbb{Z}_q^n$, we can efficiently find a $\underbrace{\text{"short"}}_{\text{in fact, } \|u\|_\infty = 1}$ $u \in \mathbb{Z}_q^m$ such that $Gu = v$!

namely SIS is easy for G.

$$u = \begin{pmatrix} v_{1,1} \\ \vdots \\ v_{1,m} \\ v_{2,1} \\ \vdots \\ v_{2,m} \\ \vdots \\ v_{n,1} \\ \vdots \\ v_{n,m} \end{pmatrix} \begin{matrix} \} \text{ binary decomposition of } v_1 \quad v_1 = \sum_{i \in [m]} 2^i \cdot v_{1,i} \\ \} \text{ binary decomposition of } v_2 \\ \\ \} \text{ binary decomposition of } v_n \end{matrix} \implies Gu = \begin{pmatrix} \sum_{i\in[m]} 2^i v_{1,i} \\ \sum_{i\in[m]} 2^i v_{2,i} \\ \vdots \\ \sum_{i\in[m]} 2^i v_{n,i} \end{pmatrix} = z$$

$\boxed{\text{Moreover, } \|u\|_\infty = 1}$

In general, for a vector $v \in \mathbb{Z}_q^n$, we write $G^{-1}(v)$ to denote the vector $u \in \mathbb{Z}_q^m$ consisting of the binary decomposition of the components of v. More generally, if we have a matrix $V \in \mathbb{Z}_q^{n \times m}$, we write $G^{-1}(V)$ to denote applying the binary decomposition operator to each column of V. Thus, we can formally define G^{-1} as the following mapping:

$$G^{-1} : \mathbb{Z}_q^{n \times m} \to \mathbb{Z}_q^{m \times m}$$

[Important: G^{-1} is not the matrix inverse of G (G is not even square)]

The matrix $G \in \mathbb{Z}_q^{n \times m}$ and the inverse mapping G^{-1} satisfy the following properties:

1. For all $V \in \mathbb{Z}_q^{n \times m}$, $G \cdot G^{-1}(V) = V$
2. For all $V \in \mathbb{Z}_q^{n \times m}$, $\|G^{-1}(V)\|_\infty = 1$

Why is this useful? Recall previous issue with multiplication: multiplying two Regev ciphertexts C_1 and C_2 causes the error in C_1 to be scaled by C_2 and C_2 is not short.

Key idea: instead of multiplying by C_2 which is big, we instead multiply by $G^{-1}(C_2)$, which is short. To recover correctness, we will use the property that $G \cdot G^{-1}(C_2) = C_2$

The GSW Homomorphic Encryption Scheme:

KeyGen(1^λ): $\tilde{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ $\quad A = \begin{bmatrix} \tilde{A} \\ \tilde{s}^T \tilde{A} + e^T \end{bmatrix} \in \mathbb{Z}_q^{(n+1) \times m}$ $\quad$ Identical to Regev encryption!

$\tilde{s} \xleftarrow{R} \mathbb{Z}_q^n$

$e \xleftarrow{R} \chi^m$ $\quad s = \begin{bmatrix} -\tilde{s} \\ 1 \end{bmatrix} \in \mathbb{Z}_q^{n+1}$

output $pk = A$ and $sk = s$

Encrypt(pk, x): Write $pk = A \in \mathbb{Z}_q^{(n+1)\times m}$ and sample $R \xleftarrow{R} \{0,1\}^{m \times m}$

$$C = AR + x \cdot G$$

[use the gadget matrix in place of the scaled identity]

Decrypt(sk, C): Write $sk = s$. Compute $s^T C$ and output 0 if $|(s^T C)_m| < \frac{q}{4}$ and 1 if $|(s^T C)_m| > \frac{q}{4}$

last component is scaled by $2^{\lfloor \log q \rfloor}$ so correctness holds as long as $B \ll q$

GSW invariant: let $C = AR + x \cdot G$ for some $x \in \{0,1\}$. Then,

$$s^T C = s^T(AR + x \cdot G) = -e^T R + x \cdot s^T G$$

last components of $s^T G$ is large ($\sim 2^{\lfloor \log q \rfloor}$) if $x = 1$ and small if $x = 0$

Homomorphic addition: $s^T(C_1 + C_2) = s^T(AR_1 + x_1 G) + s^T(AR_2 + x_2 \cdot G) = \underbrace{-e^T(R_1 + R_2)} + (x_1 + x_2) \cdot s^T G$

new error in ciphertext also adds

Homomorphic multiplication: $s^T(C_1 G^{-1}(C_2)) = s^T(AR_1 + x_1 G) \cdot G^{-1}(C_2) = s^T(AR_1 G^{-1}(C_2) + x_1 \cdot C_2)$

$$= s^T(AR_1 G^{-1}(C_2) + x_1 AR_2 + x_1 x_2 \cdot G)$$

$$= -e^T \underbrace{(R_1 G^{-1}(C_2) + x_1 R_2)} + x_1 x_2 \cdot s^T G$$

new error only increases modestly since $x_1 \in \{0,1\}$ and $G^{-1}(C_2)$ is short: if $\|R_1\|_\infty, \|R_2\|_\infty \leq B$, then

$$\|R_1 G^{-1}(C_2) + x_1 R_2\|_\infty \leq B(m+1)$$

Conclusion: If we want to support circuits of multiplicative depth d, we need to choose $q = m^{O(d)}$ to accomodate the multiplications. Observe that in this case, $\log q = O(d \log m)$, so the number of bits in the ciphertext scales linearly with the depth of the circuit. [Note: generally, there is a lot of flexibility when choosing lattice parameters]

Semantic security follows by same argument as Regev. Homomorphic operations possible by structure of gadget matrix!

From SWHE to FHE. The above construction requires imposing an a priori bound on the multiplicative depth of the computation. To obtain fully homomorphic encryption, we apply Gentry's brilliant insight of bootstrapping.

High-level idea. Suppose we have SWHE with following properties:

1. We can evaluate functions with multiplicative depth d
2. The decryption function can be implemented by a circuit with multiplicative depth $d' < d$

Then, we can build an FHE scheme as follows:

- Public key of FHE scheme is public key of SWHE scheme and an encryption of the SWHE decryption key under the SWHE public key
- We now describe a ciphertext-refreshing procedure:
  - For each SWHE ciphertext, we can associate a "noise" level that keeps track of how many more homomorphic operations can be performed on the ciphertext (while maintaining correctness).
    - ↳ for instance, we can evaluate depth-d circuits on fresh ciphertexts; after evaluating a single multiplication, we can only evaluate circuits of depth-$(d-1)$ and so on...
  - The refresh procedure takes any valid ciphertext and produces one that supports depth-$(d-d')$ homomorphism; since $d > d'$, this enables unbounded (i.e., arbitrary) computations on ciphertexts

Idea: Suppose $ct_x = \text{Encrypt}(pk, x)$.

Using the SWHE, we can compute $ct_{f(x)} = \text{Encrypt}(pk, f(x))$ for any f with multiplicative depth up to d

Given ct_x, we first compute

$$ct_{ct} = \text{Encrypt}(pk, ct_x) \quad \text{[strictly speaking, encrypt bit by bit]}$$

This is a fresh ciphertext so we can perform operations of depth up to d on ct_{ct}. Since the public key includes a copy of the decryption key (ct_{sk}), we can homomorphically evaluate the decryption function:

$$\left.\begin{array}{l} ct_{ct} = \text{Encrypt}(pk, ct_x) \\ ct_{sk} = \text{Encrypt}(pk, sk) \end{array}\right\} \underbrace{\text{Encrypt}(pk, \text{Decrypt}(sk, ct))}_{\text{depth-}d'\text{ computation}} = \text{Encrypt}(pk, x)$$

This is a new encryption of m, and we can continue performing homomorphic operations on m (of depth $d - d'$)

Bootstrapping is a general technique that converts any SWHE that can evaluate its own decryption function (plus a little more) into an FHE scheme. Transformation requires additional circular security assumption (namely, that it is OK to publish an encryption of the scheme's own public key. [The GSW scheme supports bootstrapping — decryption is a threshold inner product; choose parameters carefully]

Open problem: Build FHE from LWE (or another standard assumption) without the circular security assumption.