Lecture 16:

Fully Homomorphic Encryption

CS 355 – Spring 2019

May 22, 2019

Henry Corrigan-Gibbs

Logistics

* HW4 Due Friday, May 24 at 5pm
* Vote for topics of bonus lectures
* HW5 out now.
* Events:
  - Grubbs talk (5/24, 4:15pm in Gates 463A)
  - Aloni Cohen (5/29, 12pm in Gates 463A) *
  - Defense (5/31, 1pm in Packard 101)
  - David Lazar (6/4, 4:15pm in Gates 463A)
  - Michael Duff (6/5, 12pm in Gates 463A) *

* = Lunch!

Plan

* Reminder of LWE
* Defn of FHE
* Leveled HE (GSW)
  - ↳ Construction
* Bootstrapping to FHE

Recap: Learning w/ Errors

NEW ASSUMPTION

LWE(n, m, q, χ_B) ... see diagram on next page

$$\left\{ (A, s^T A + e^T) \;\middle|\; \begin{array}{l} A \xleftarrow{R} \mathbb{Z}_q^{n \times m} \\ s \xleftarrow{R} \mathbb{Z}_q \\ e \xleftarrow{R} \chi_B \end{array} \right\} \approx_c \left\{ U \;\middle|\; U \leftarrow \mathbb{Z}_q^{(n+1) \times m} \right\}$$

Solving noisy overdetermined system of eqns is hard.

Regev Encryption

- PKE w/ sem. sec under LWE assumption
- Same ideas used to construct FHE scheme (today).

Recap : LWE

$\lambda = 128$

$n \approx 600 - 800$

$m = n \lg q$

$q = n^2 \sim 2^{\sqrt{n}}$

LWE assumption (params n, m, q, χ)

for

$$A \overset{R}{\leftarrow} \mathbb{Z}_q^{n \times m}$$

$$s \overset{R}{\leftarrow} \mathbb{Z}_q^{n}$$

$$e^T \overset{R}{\leftarrow} \chi$$

$$\left\{ \begin{matrix} A \\ s^T A + e^T \end{matrix} \right\} \approx_c \left\{ \text{Uniform in } \mathbb{Z}_q^{(n+1) \times m} \right\}$$

Fully Homomorphic Encryption (FHE)

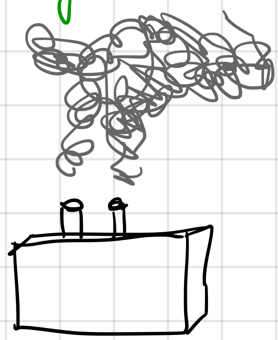
Idea: Outsource computation without revealing inputs.

Data → x

Enc(x) →

← Enc($f(x)$)

↓ $f(x)$



Map
Enc(x) ↦ Enc($f(x)$)

Examples:

* Private lookup (PIR)
  - Enc(i) →
  - ← Enc(i-th row of DB)
* Private spam filtering
  - Enc(email msg) →
  - ← Enc($\begin{cases} 1 & \text{if spam} \\ 0 & \text{o.w.} \end{cases}$)
* Private ML
  - Enc(data set) →
  - ← Enc(ML model trained on data set)
* And so on....

N.B. FHE alone isn't powerful enough to do everything we'd like to do (e.g. secure comp on joint inputs), but it's still <u>very</u> powerful.

N.B. These schemes <u>aren't</u> used in practice yet...

Some history...

1978 - Rivest, Adleman, Dertouzos (1978) introduce notion of FHE (under other name)

↳ We've had partially hom. enc. for a long time (RSA, Paillier, ElGamal, ...)

↳ No unbroken candidates

2009 - Craig Gentry (Stanford PhD student in Dan's group) gives first candidate FHE scheme

I think it's fair to call this a major breakthrough

↳ From new assumption (non-standard)

↳ Introduced crucial "bootstrapping" idea

2011 - Schemes based on LWE (Brakerski, Vaikuntanathan)
+ "circular-security assumption"

2013 - "Third-gen scheme" that we'll see today (Gentry, Sahai, Waters)

Syntax of FHE (Based on notes from Sam Kim.)

$\text{KeyGen}(1^\lambda) \to sk$

$\text{Enc}(sk, \mu) \to ct$

$\text{Eval}(C, ct_1, \ldots, ct_\ell) \to \tilde{ct}$

$\text{Dec}(sk, ct) \to \mu$

(Can define a PKE notion of FHE as well...)

Note: The computation is represented by a boolean ckt. This is the source of some of the ineff. in practice, but is wlog wrt polynomial-time comps.

Properties

1. Correctness. Let $C: \{0,1\}^\ell \to \{0,1\}$

$\forall sk$ output by KeyGen, $\forall C$, $\forall \mu_1, \ldots, \mu_\ell$

$$\mathrm{Dec}(sk, \mathrm{Eval}(C, \mathrm{Enc}(sk, \mu_1), \ldots, \mathrm{Enc}(sk, \mu_\ell))) = C(\mu_1, \ldots, \mu_\ell).$$

2. Semantic Security.

$\forall sk$ output by KeyGen, $\forall \mu_0, \mu_1$

$$\{\mathrm{Enc}(sk, \mu_0)\} \approx_c \{\mathrm{Enc}(sk, \mu_1)\}$$

3. Compactness. Dec time is indep of $|C|$.

$\forall sk$

$\forall C$, $ct_1, \ldots, ct_\ell$ output by $\mathrm{Enc}(sk, \cdot)$

if $\widetilde{ct} \leftarrow \mathrm{Eval}(C, ct_1, \ldots, ct_\ell)$

then $|\widetilde{ct}| = \mathrm{poly}(\lambda)$ ← independent of $|C|, \ell$

Claim: w/o compactness, any sem. sec. enc scheme is also an FHE.

Idea behind FHE

All FHE schemes we have use a two-step approach

1. Construct "leveled" HE
   - ↳ Eval handles ckts of bounded depth.
   - ↳ Intuition: There's some noise/error in cts that grows with each * gate in ckt. Eventually, error gets too large and breaks correctness.

2. Use bootstrapping to convert leveled HE → FHE
   - ↳ This is the mind-blowing idea from Gentry (2009) ... so simple, so nice

↳ Idea: Publish $ct_{sk} = \mathrm{Enc}(pk, sk)$ as part of public info. (Think of a public-key FHE ...)

Compute $ct_{ct} \leftarrow \mathrm{Enc}(pk, ct_\mu)$ ("noisy" encryption of μ)

$\hat{ct} \leftarrow \mathrm{Eval}(\mathrm{Dec}, ct_{sk}, ct_{ct})$ ("Clean" encryption of ct_μ; Eval(Dec, ...): Homomorphically decrypt the noisy ct_μ)

$\hat{ct} = \mathrm{Enc}(pk, \mathrm{Dec}(sk, ct))$

$= \mathrm{Enc}(sk, \mu)$ — "clean encryption of μ"

⇒ Can remove noise by bootstrapping. Can compute arbitrary ckts on encrypted data.

Digression:
Two good ideas in CS.
Both from Turing.
FHE uses both...
st excellent....

Two comments about "bootstrapping"

1. Publishing Enc(sk, sk) might be risky.

   ↳ Semantic security of Enc alone doesn't imply that this is safe.

   (Ex. Construct a sem. sec. enc scheme for which publishing Enc(sk, sk) leads to a total break of the scheme.)

   When using bootstrapping we make a "circular security" assumption --- just assume that this is safe.

2. We need to run $\mathrm{Eval}(\widehat{\mathrm{Dec}}, \ldots)$

   ↳ Our SWHE scheme must support ckts of depth $\geq$ depth($\widehat{\mathrm{Dec}}$).

   ↳ Want a SWHE scheme with a "shallow" decryption circuit.

Preliminaries

We'll use the "Gadget matrix" $G \in \mathbb{Z}_q^{n \times n \log q}$

$$G = \begin{pmatrix} 1 \; 2 \; 4 \cdots 2^{q/2} & & & \\ & 1 \; 2 \; 4 \cdots 2^{q/2} & & \\ & & \ddots & \\ & & & 1 \; 2 \; 4 \cdots 2^{q/2} \end{pmatrix}$$

Bit decomposition

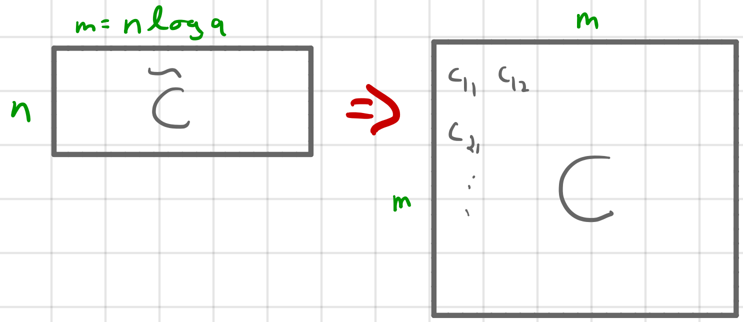
For a matrix $\tilde{C} \in \mathbb{Z}_q^{n \times n \log q}$ ($n \log q = m$), define the bit decomp operation

$$C = \text{BitDecomp}(\tilde{C}) \in \mathbb{Z}_q^{m \times m}$$

that splits each $\mathbb{Z}_q$ element x in $\tilde{C}$ into a vector

$$x \in \mathbb{Z}_q \longrightarrow \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{\log q} \end{pmatrix} \in \{0,1\}^{\log q \times 1} \subseteq \mathbb{Z}_q^{\log q \times 1}$$

Then:



Key Idea: $\tilde{C} = G \cdot \text{BitDecomp}(\tilde{C})$

Now, we construct leveled HE scheme...

GSW FHE (secret key)

← Shares ideas w/ Regev encryption

KeyGen(1^λ): $\tilde{s} \xleftarrow{R} \mathbb{Z}_q^{n-1}$

$sk = (\tilde{s} \;\; -1) \in \mathbb{Z}_q^n$

Encrypt(s, μ) ($\mu \in \{0,1\}$): $A \xleftarrow{R} \mathbb{Z}_q^{(n-1) \times n \lg q}$ ($n \lg q = m$)

$e \xleftarrow{R} \chi$

$$\tilde{C} = \underbrace{\begin{pmatrix} A \\ \tilde{s}^T A + e^T \end{pmatrix}}_{\text{pseudorandom by LWE}} + \mu \cdot G \in \mathbb{Z}_q^{n \times n \lg q}$$

($n \lg q = m$)

output $C \leftarrow \text{BitDecompose}(\tilde{C}) \in \mathbb{Z}_q^{m \times m}$

Decrypt(s, C):

Write $s^* \leftarrow s^T G \in \mathbb{Z}_q^{n \log q}$

Then $(s^*)^T C = s^T G \cdot C$ (annotations: $(s^*)^T$ is $1 \times m$; C is $m \times m$)

$$= s^T \tilde{C}$$

$$= s^T \left[\begin{pmatrix} A \\ \tilde{s}^T A + e^T \end{pmatrix} + \mu \cdot G \right]$$

$$= (\tilde{s}^T \ -1) \begin{pmatrix} A \\ \tilde{s}^T A + e^T \end{pmatrix} + \mu s^T G$$

$$= \tilde{s}^T A - (\tilde{s}^T A + e^T) + \mu s^T G$$

$$= \mu s^T G - \underbrace{e^T}_{\text{small noise}}$$

$$= \begin{cases} \text{"small"} & \text{if } \mu = 0 \quad (< q/4) \\ \text{"big"} & \text{if } \mu = 1 \quad (\geq q/4) \end{cases}$$

Now to compute hom. operations...

It's enough to show how to go from $C_{\mu_1}, C_{\mu_2} \mapsto C_{\mu_1 \text{ NAND } \mu_2}$, since NAND is a universal gate.

Homomorphic NAND:

$$C_{\mu_1}, C_{\mu_2} \longmapsto I_m - (C_{\mu_1} \cdot C_{\mu_2})$$

Then $(s^*)^T [I_m - C_{\mu_1} \cdot C_{\mu_2}]$

$$= s^T G - (s^T G C_{\mu_1}) \cdot C_{\mu_2}$$

$$= s^T G - (s^T \mu_1 G + e_1^T) C_{\mu_2}$$

$$= s^T G - \mu_1 s^T G C_{\mu_2} + e_1^T C_{\mu_2}$$

$$= s^T G - \mu_1 (\mu_2 s^T G + e_2^T) + e_1^T C_{\mu_2}$$

$$= s^T G - \mu_1 \mu_2 s^T G \underbrace{- \mu_1 e_2^T + e_1^T C_{\mu_2}}$$

$$= (1 - \mu_1 \mu_2) s^T G + \langle \text{noise} \rangle$$

Small noise since $|\mu_1| \leq 1$, $\|C_{\mu_2}\|_\infty \leq 1$

Large if μ_1 NAND $\mu_2 = 1$

Small if μ_1 NAND $\mu_2 = 0$

The catch...

With each homomorphic NAND we perform, the noise grows.

↳ If we perform many nands, noise will overwhelm the "signal".

After computing a ckt of depth L, noise will be bounded by

$$(m+1)^{depth} \cdot B$$

← original noise bound

We need $(m+1)^{depth} \cdot B \ll q$

$$(m+1)^{depth} \ll q/B$$

$$depth \cdot \lg(m+1) \ll \log(q/B)$$

$$depth \ll \lg(q/B) / \lg(m+1)$$

↳ If we take $q \approx 2^{\sqrt{n}}$, can handle ckts of lg depth.

↳ Regev-style encryption has a decryption ckt of $O(\lg n)$ depth... it's just a matrix-vector product plus rounding.

↳ We can support bootstrapping, and we're done!

When will this be practical?