Computing with Encrypted Data

6.857 Lecture 26
Encryption for Secure Communication

- All-or-nothing
  - Have Private Key, Can Decrypt
  - No Private Key, No Go
    - cf. Non-malleable Encryption
Encryption for Cloud Computing

Data Analysis & Statistics, Classification, Search, Image Processing, ...

Cloud

Compute Function F

Enc(F(Data))

Medical, Financial and other Personal Information

Need: Privacy + Functionality
Fully Homomorphic Encryption

Compute arbitrary functions on encrypted data?

Enc(data),  F → Enc(F(data))

(fully = any function F)
(additive = only additions)
(multiplicative = only mult)
(somewhat = circuits of small depth)

[Rivest, Adleman and Dertouzos’78]

[ON DATA BANKS AND PRIVACY HOMOMORPHISMS]

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos

Massachusetts Institute of Technology
Cambridge, Massachusetts

[Gentry’09, BV’11, LTV’12]: Constructions of FHE
Outline

◆ Homomorphic Encryption
  – Multiplicative Homomorphism: El Gamal
  – Additive Homomorphism: Goldwasser-Micali
  – Fully Homomorphic Encryption: based on NTRU

◆ What I didn’t tell you (and how to learn more)
If we had:

• Enc(x₁), Enc(x₂) ⇒ Enc(x₁ + x₂)
• Enc(x₁), Enc(x₂) ⇒ Enc(x₁ · x₂)

then we are done.
Multiplicative Homomorphism

El Gamal Encryption

Setup: Group G of prime order p
(e.g., set of quadratic residues mod q where q = 2p+1)

Private key: \( x \in \mathbb{Z}_p \)

Public key: generator \( g \), \( y = g^x \in G \)

\[ \text{Enc}(m_1): \ (g^{r_1}, y^{r_1} \cdot m_1) \]

\[ \text{Dec}(m): \text{Observe that } (g^{r_1})^x = y^{r_1} \]
**Multiplicative Homomorphism**

<table>
<thead>
<tr>
<th>El Gamal Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup:</strong> Group $G$ of prime order $p$</td>
</tr>
<tr>
<td>(e.g., set of quadratic residues mod $q$ where $q = 2p+1$)</td>
</tr>
<tr>
<td><strong>Private key:</strong> $x \in \mathbb{Z}_p$</td>
</tr>
<tr>
<td><strong>Public key:</strong> generator $g$, $y = g^x \in G$</td>
</tr>
<tr>
<td>$\text{Enc}(m_1)$: $(g^{r_1}, y^{r_1} \cdot m_1)$</td>
</tr>
<tr>
<td>$\text{Enc}(m_2)$: $(g^{r_2}, y^{r_2} \cdot m_2)$</td>
</tr>
<tr>
<td>$\overline{(g^{r_1+r_2}, y^{r_1+r_2} \cdot m_1m_2)}$</td>
</tr>
</tbody>
</table>

is an encryption of the product $m_1m_2$
Additive Homomorphism

Goldwasser Micali Encryption

Public key: $N$, $y$: non-square mod $N$

Secret key: factorization of $N$

$\text{Enc}(0): r^2 \mod N, \quad \text{Enc}(1): y \cdot r^2 \mod N$

\[
\begin{align*}
\text{square (0) } \cdot \text{ square (0)} &= \text{ square (0)} \\
\text{non-square (1) } \cdot \text{ square (0)} &= \text{ non-square (1)} \\
\text{square (0) } \cdot \text{ non-square (1)} &= \text{ square (1)} \\
\text{non-square (1) } \cdot \text{ non-square (1)} &= \text{ non-square (0)}
\end{align*}
\]

XOR-homomorphic: Just multiply the ciphertexts
Other HE Schemes

• **Additively Homomorphic:**
  – Paillier
  – Damgard-Jurik (both addition of large numbers)

• **Additions + a single Multiplication:**
  – Boneh-Goh-Nissim (based on gap groups)
  – Gentry-Halevi-V. (based on lattices)

• **HE with Large ciphertext blowup:**
  – Sander-Young-Yung
How to Construct an FHE Scheme
The Big Picture

STEP 1

“Somewhat Homomorphic” (SwHE) Encryption
[Gen09, DGHV10, SV10, BV11a, BV11b, BGV12, LTV12, GHS’12]

Evaluate arithmetic circuits of depth $d = \varepsilon \log n$ *

* (0 < $\varepsilon$ < 1 is a constant, and $n$ is the security parameter)
The Big Picture

STEP 2

“Bootstrapping” Theorem [Gen09] (Qualitative)

“Homomorphic enough” Encryption $\Rightarrow^*$ FHE

Homomorphic enough = Can evaluate its own Dec Circuit (plus some)

Decryption Circuit

$$\epsilon^{(\cdot)}$$

EVAL
The Big Picture

STEP 1

“Somewhat Homomorphic” (SwHE) Encryption
[Gen09, DGHV10, SV10, BV11a, BV11b, BV12, LTV12, GHS’12]

Evaluate arithmetic circuits of depth $d = \varepsilon \log n$

STEP 2

“Bootstrapping” Method

“Homomorphic enough” Encryption $\Rightarrow$* FHE

Homomorphic enough =
Can evaluate its own Dec Circuit (plus some)

STEP 3

Depth Boosting / Modulus Reduction [BV11b]
Boost the SwHE to depth $d = n^\varepsilon$
The NTRU Encryption Scheme
[Hofstein-Pipher-Silverman’97]

Central characters: Polynomials mod \( q \)

- Polynomials of degree less than \( n \) (think \( n = 256 \))
- Coefficients over \( \mathbb{Z}_q \) (think \( q = \) small prime)

- Addition: coefficient-wise
  \[
  (6x^2+5x+10) + (5x^2+x+2) = 6x+1 \quad \text{(mod 11)}
  \]

- Multiplication: polynomial mult, modulo an irreducible
  \[
  (6x^2+5x+10) \times (5x^2+x+2) = 9x^3+x^2+9x+1
  \quad \text{(mod 11, } x^4+1)\]

Ring \( R_q := \mathbb{Z}_q[x] / (x^n+1) \) \( (x^n+1 \text{ cyclotomic, } q = 1 \text{ mod } 2n \text{ prime}) \)
The NTRU Encryption Scheme

• **KeyGen:**
  - Sample “small” polynomials \( f, g \in \mathbb{R}_q \) (s.t. \( f=1 \text{ mod } 2 \))
  - Secret key \( SK=f \) and Public key \( PK=h=2g/f \)

• **Encryption** \( Enc_{pk}(m) \):
  - Sample “small” polynomials \( s, e \in \mathbb{R}_q \),
  - output \( C = hs + 2e + m \pmod{q, x^{n+1}} \)

• **Decryption** \( Dec_{sk}(C) \): Output \( fC \pmod{q,x^{n+1}} \) mod 2.

  • **Correctness:** \( fC = f(hs+2e+m) = 2(gs+fe) + fm \pmod{q, x^{n+1}} \)
    
    If \( |2(gs+fe) + fm| < q/2 \), taking mod 2 gives \( m \).
The NTRU Encryption Scheme

The "Small Polynomial Ratios" (SPR) Assumption:
Choose two polynomials $f$ and $g$ with "small" coefficients (of magnitude at most $B$). Then,

$$\frac{g}{f} \approx_c \text{uniform over } R_q$$

The key security parameter: The signal-to-noise ratio $q/B$

If $q/B$ is too large ($> 2^n$), we can break NTRU in poly time.

Therefore, typical setting: $q/B = 2^{n^\varepsilon}$ (for some $\varepsilon << 1$)

Theorem: The encryption scheme is secure under the SPR assumption
Additive Homomorphism

\[ c_1 = h s_1 + 2e_1 + m_1 \]
\[ f.c_1 = 2E_1 + f m_1 \]

\[ c_2 = h s_2 + 2e_2 + m_2 \]
\[ f.c_2 = 2E_2 + f m_2 \]

Add the ciphertexts:
\[ c_{\text{add}} = c_1 + c_2 \text{ (over } R_q \text{)} \]

\[
\begin{align*}
& f.c_1 = 2E_1 + fm_1 \\
+ & f.c_2 = 2E_2 + fm_2 \\
& f.(c_1+c_2) = 2(E_1+E_2) + f.(m_1+m_2) \\
\Rightarrow & \text{Dec}_f(c_{\text{add}}) = 2E' + f.(m_1+m_2) \text{ (mod 2)} = f. (m_1+m_2) \text{ (mod 2)} \\
& = m_1 + m_2 \text{ (mod 2)}
\end{align*}
\]
## Multiplicative Homomorphism

<table>
<thead>
<tr>
<th>$c_1 = hs_1 + 2e_1 + m_1$</th>
<th>$c_2 = hs_2 + 2e_2 + m_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f.c_1 = 2E_1 + fm_1$</td>
<td>$f.c_2 = 2E_2 + fm_2$</td>
</tr>
</tbody>
</table>

Multiply the ciphertexts: $c_{\text{mlt}} = c_1.c_2$ (over $R_q$)

$$
\begin{align*}
    f.c_1 &= 2E_1 + fm_1 \\
    \times \\
    f.c_2 &= 2E_2 + fm_2 \\
    f^2.(c_1c_2) &= 2(E_1m_2 + E_2m_1 + 2E_1E_2) + f^2(m_1m_2) \\
    \Rightarrow \text{Dec}_{f^2}(c_{\text{mlt}}) &= 2E' + f^2m_1m_2 \pmod{2} = f^2m_1m_2 \pmod{2} = m_1m_2 \pmod{2}
\end{align*}
$$
Noise Growth

- Assume the input ciphertext noise is at most $B$.

- Addition: norm of $E_1 + E_2$ is at most $2B$

- Multiplication: noise $\approx E_1 E_2$
  - Norm of $E_1 E_2$ is at most $nB^2$
How Homomorphic is this:
The Reservoir Analogy

- **Additive Homomorphism:** $B \rightarrow 2B$

- **Multiplicative Homomorphism:** $B \rightarrow nB^2$

**AFTER d LEVELS:**

- Noise: $B \rightarrow (nB)^{2^d}$ (worst case)

**Correctness**

- SPR with $q/B$ ratio $2^{n^\varepsilon}$
How Homomorphic is this:
The Reservoir Analogy

Additive Homomorphism: $B \rightarrow 2B$

Multiplicative Homomorphism: $B \rightarrow nB^2$

AFTER $d$ LEVELS:

$\text{noise } B \rightarrow (nB)^{2d}$ (worst case)

\[
(nB)^{2d} \leq \frac{q}{2} \leq B \cdot 2^{n\varepsilon}
\]

\[
d \leq \log(\log q) - \log(\log nB) \\
\leq \varepsilon \log n - \log \log n
\]
“Somewhat Homomorphic” (SwHE) Encryption

Evaluate arithmetic circuits of depth $d = \varepsilon \log n$

Depth Boosting / Modulus Reduction

Boost the SwHE to depth $d = n^\varepsilon$

“Bootstrapping” Method

“Homomorphic enough” Encryption $\Rightarrow^*\text{ FHE}$

Homomorphic enough = Can evaluate its own Dec Circuit (plus some)
Bootstrapping Theorem [Gen09]

- If you can homomorphically evaluate depth $d$ circuits and
- the depth of your decryption circuit $< d$

$\Rightarrow$ FHE
Bootstrapping

Bootstrapping Theorem [Gen09]

\[ d\text{-HE with decryption depth} < d \implies^\ast \text{FHE} \]

Bootstrapping = “Valve” at a fixed height
(that depends on decryption depth)

\[ \text{Say } n(B_{\text{dec}})^2 < q/2 \]
Bootstrapping

**Bootstrapping Theorem** [Gen09]

\[ d\text{-HE with decryption depth } < d \implies^* \text{ FHE} \]

Bootstrapping = “Valve” at a fixed height
(that depends on decryption depth)

\[
\begin{align*}
\text{noise} &= q/2 \\
\text{noise} &= B_{\text{dec}} \\
\text{noise} &= 0
\end{align*}
\]

Say \( n(B_{\text{dec}})^2 < q/2 \)
But the evaluator (cloud) does not have SK!

“Best Possible” noise reduction = Decryption!

Decryption Circuit

“Very Noisy” ciphertext $\rightarrow$ $CT$ $\rightarrow$ $SK$

“Noiseless ciphertext”
Bootstrapping, Concretely

Next Best = Homomorphic Decryption!

* Assume $\text{Enc}(SK)$ is public.
(OK assuming the scheme is “circular secure”)

$\text{Enc}_{PK}(m)$

Noise = $B_{\text{dec}}$

$B_{\text{dec}}$ Independent of $B_{\text{input}}$

Noise = $B_{\text{input}}$
Wrap Up: Bootstrapping

Assume Circular Security:
Public key contains $\text{Enc}_{SK}(SK)$
Wrap Up: Bootstrapping

Assume Circular Security:
Public key contains $\text{Enc}_{sk}(SK)$

Each Gate $g \rightarrow$ Gadget $G$:

$$g(a,b)$$

$$\text{Dec}$$

$$a$$

$$c_a$$

$$\text{sk}$$

$$b$$

$$c_b$$

$$\text{sk}$$
Assume Circular Security:
Public key contains $Enc_{SK}(SK)$

Each Gate $g \rightarrow$ Gadget $G$:

Wrap Up: Bootstrapping

Function $f$

$Enc(g(a,b))$

Dec

Dec

$c_a$ $Enc(SK)$ $c_b$ $Enc(SK)$
Wrap-up: FHE

**STEP 1**

“Somewhat Homomorphic” (SwHE) Encryption

[Gen09, DGHV10, SV10, BV11a, BV11b, BGV12, LTV12, GHS’12]

Evaluate arithmetic circuits of depth \( d = \varepsilon \log n \)

**STEP 2**

“Bootstrapping” Method

“Homomorphic enough” Encryption \( \Rightarrow^* \) FHE

Homomorphic enough =
Can evaluate its own Dec Circuit (plus some)

**STEP 3**

Depth Boosting / Modulus Reduction [BV11b]

Boost the SwHE to depth \( d = n^\varepsilon \)
Boosting Depth from log n to $n^\varepsilon$
(in one slide)

- The culprit: Multiplication
- Increases noise from B to $nB^2 \gg B$

- Let us pause for a moment. Is $nB^2 > B$?

- … Not if $B < 1$

- Why not scale everything by q, and work over $(0,1)$?  
- Quite amazingly, this works out and gives us an error growth of $nB$ (no squaring)
**Wrap-up: FHE**

| STEP 1 | “Somewhat Homomorphic” (SwHE) Encryption
[Gen09,DGHV10,SV10,BV11a,BV11b,BGV12,LTV12,GHS’12]  
Evaluate arithmetic circuits of depth $d = \varepsilon \log n$ |
|---|---|
| STEP 2 | “Bootstrapping” Method  
“Homomorphic enough” Encryption $\Rightarrow^* FHE$  
Homomorphic enough = Can evaluate its own Dec Circuit (plus some) |
| STEP 3 | Depth Boosting / Modulus Reduction [BV11b]  
Boost the SwHE to depth $d = n^\varepsilon$ |
What We Didn’t Do

A Lot!

- Functional Encryption
- Software Obfuscation: how to encrypt programs
- Practical techniques for computing on encrypted data: searchable encryption, deterministic encryption,…
- Secure Multiparty protocols, …

Come to 6.892!
Thanks!

Good luck with the project write-ups!