Fully Homomorphic Encryption from the ground up

Daniele Micciancio
(UC San Diego)

Eurocrypt 2019
(Fully Homomorphic) Encryption

• Encryption: used to protect data at rest or in transit
  
  \[ \text{Enc}(m) \]

• Fully Homomorphic Encryption: supports arbitrary computations on encrypted data
  
  \[ \text{Enc}(m) \]
  \[ \text{Enc}(F(m)) \]
FHE Timeline

- Concept originally proposed by Rivest, Adleman, Dertouzos (1978)
- Gentry’s breakthrough (2009)
  - First candidate solution
  - Bootstrapping technique
- Much subsequent work (2010-2019 …)
  - Basing security on standard (lattice) assumptions [BV11,B12,AP13,GSW13,BV14,…]
  - Efficiency improvements [GHS12,BGH13,AP13/14,DM15,CP16,CGGI16/17,CKKS17,MS18,…]
  - Implementations: HElib, SEAL, PALISADE, FHEW, TFHE, HeaAn, Λολ, NFLlib, …
Outline

• FHE: background and sample applications
• Lattice Cryptography
  – Key properties of lattice cryptography that make it so useful to build FHE and other applications
• Generic FHE construction
  – Symmetric Encryption
  – Public Key Encryption
  – Linearly Homomorphic Encryption
  – Fully Homomorphic Encryption
FHE applications

- Direct applications:
  - Secure outsourcing of computation
- Powerful tool: “Cryptographic Pantograph”
  - FHE [Gentry09]
  - (Indistinguishability) Obfuscation [GGHRSW13]
  - Functional Encryption [GKPVZ13]
  - Correlation Intractable Hash Functions [PS19], [CCHLRRW19]
Sample Application 1

- **(Indistinguishability) Obfuscation**
  - Obf: Program $\rightarrow$ Program
  - Correctness: $\text{Obf}[P](x) = P(x)$
  - Security: $P_0(x) = P_1(x) \rightarrow \text{Obf}[P_0] \sim \text{Obf}[P_1]$
Bootstrap Obfuscation

- Bootstrapping Obfuscation using FHE
  - Obf’: obfuscation scheme for simple/small P’
- Obf[P] = (Enc(P),Obf’[Dec(.)])
  - (Enc,Dec,Eval)←FHE.KeyGen
- Obf[P](x) = Dec(e)
  - Obf’[Dec(.)] (Eval(Enc(P),x))
    = Dec(Enc(P(x))) = P(x)

- Actual scheme is a bit more complex:
  - encrypt/evaluate P twice, under two different FHE keys
  - check consistency before decryption
Sample Application 2

- **Correlation Intractable Hash Functions**
  - Hash function $H(x)$, Relation $R = \{(x,f(x)) : x\}$
  - Security: Hard to find $x$ such that $R(x,H(x))$
- $H=“Random oracle”$ is “trivially” secure

- Applications:
  - Fiat-Shamir Signatures in the Standard Model
  - Remove interaction in public coin protocols
  - Non-Interactive Zero-Knowledge
Bootstrapping Correlation Intractability

- $H'$: CI Hash function for simple relation
  \[ R(x,y) = "y=\text{Dec}(x)" \]
  for some $\text{Dec} \leftarrow \text{FHE.KeyGen}$

- $H$: CI Hash function for arbitrary $P$
  - $(\text{Enc},\text{Dec},\text{Eval}) \leftarrow \text{FHE.KeyGen}$
  - $C = \text{Enc}(P)$
  - $H(x) = H'(\text{Eval}(C,x))$

- Security:
  - Assume $H(x) = P(x)$
  - Let $c = \text{Eval}(C,x) = \text{Enc}(P(x))$
  - Then $H'(c) = H(x) = P(x) = \text{Dec}(c)$
Lattice cryptography

- Lattices: regular sets of vectors in n-dim space

- Many attractive features:
  - Post-Quantum secure candidate
  - Simple, fast and easy to parallelize
  - Versatile (FHE and much more)
Why Lattice Cryptography?

• Lattices → Encryption
  - weak linear homomorphic properties
  - simple (linear) decryption algorithm
  - circular secure: $\text{Enc}_s(s)$ does not leak $s$

• This is enough to obtain
  - multiplication by arbitrary constants
  - multiplications between ciphertexts
  - fully homomorphic encryption
Learning With Errors (LWE)

- **LWE function family:**
  - **Key:** \( A \in \mathbb{Z}_q^{nxm} \)
  - \( \text{LWE}_A(s,e) = As + e \pmod{q} \)
  - Small \(|e| \leq \beta = O(\sqrt{n})\)
  - \( q, m = \text{poly}(n) \)
  - Injective version of Ajtai’s SIS function

- **Regev (2005): assuming quantum hard lattice problems**
  - \( \text{LWE}_A \) is one-way: Hard to recover \((s,e)\) from \([A,b]\)
  - \( b = \text{LWE}_A(s,e) \) is indistinguishable from uniform over \( \mathbb{Z}_q^m \)
  - [BLPRS13] hard under classical reductions
Encrypting with LWE

- Idea: Use $b=LWE_A(s,e)$ as a one-time pad
- Private key encryption scheme:
  - secret key: $s \in \mathbb{Z}_q^n$,
  - message: $m \in \mathbb{Z}^m$
  - encryption randomness: $[A,e]$
  - $E_s(m; [A,e]) = [A,b+m]$

- [BFKL93],[GRS08]
  - Learning Parity with Noise (LPN): $q=2$
  - If $LWE_A$ is one-way, then $b=As+e$ is pseudo-random
- Regev LWE: $q \rightarrow \text{poly}(n)$
Noisy Decryption

• $E_s(m;[A,e]) = [A,b+m]$ where $b = As + e$

• Decryption:
  – $D_s([A,b+m]) = (b+m) - As = m + e \mod q$

• Low order bits of $m$ are corrupted by $e$

• Fix: scale $m$, and round:
Weak Linear Homomorphism

- \([A_1, A_1s + e_1 + m_1] + [A_2, A_2s + e_2 + m_2]\]
  \[= [(A_1 + A_2), (A_1 + A_2)s + (e_1 + e_2) + (m_1 + m_2)]\]

\(E_s(m; \beta)\): encryption of m with error \(|e| < \beta\)

- \(E_s(m_1; \beta_1) + E_s(m_2; \beta_2) \subseteq E_s(m_1 + m_2; \beta_1 + \beta_2)\)
Circular Security

- $E_s(m; [A,e]) = [A,b+m]$, where $b = As + e$
- $D_s([A,b+m]) = (b+m) - As = m + e$
- $D_s([-A,0]) = 0 + As = As$

- Easy to compute encryptions of (linear functions of) the secret key $s$!

- Random encryptions:
  
  $[-A,0] + E_s(0; \beta) = E_s(As; \beta)$
Decryption is also linear

- $D_s(A,b) = b - As = m + e$
- Linear in the ciphertext $(A,b)$
- Linear in the secret key $s' = (-s, 1)$
  - $D_{s'}(A,b) = [A,b]s' = m + e$
  - $D_{cs'}(A,b) = [A,b](cs') = cm + ce$
- Remark:
  - Only approx. decryption is linear
  - Exact decryption involves non-linear rounding
Operations on Ciphertexts

- **Add:** \( E(m_1; \beta_1) + E(m_2; \beta_2) \subseteq E(m_1 + m_2; \beta_1 + \beta_2) \)
- **Neg:** \(-E(m; \beta) = E(-m; \beta)\)
- **Mul:** \( c \cdot E(m; \beta) = E(c \cdot m; c \cdot \beta) \)
- **Const:** \( [0, m] \in E(m; 0) \)
- **Key:** \( [-A, 0] \in E(A \cdot s; 0) \)

**Weak linear homomorphic properties:**
- can perform a limited number of additions and multiplications by small constants
- decryption is linear in the secret key \( s' = (-s, 1) \)
- circular security: \( E(A \cdot s) \) does not leak \( s \)
Public Key Encryption

- Public Key:
  \[[a_1, b_1] = E_s(0), \ldots, [a_n, b_n] = E_s(0)\]

- Encrypt(m): \((\Sigma_i r_i \cdot [a_i, b_i]) + (0, m)\)
  \(- E_s(0) + \ldots + E_s(0) + E_s(m; 0) = E_s(m)\)

- Decrypt normally using secret key

- [Regev05] LWE Public Key Encryption

- [Rothblum11]: any weakly linear homomorphic encryption implies public key encryption
Multiplication by any constant

- $E'[m] = (E[m], E[2m], E[4m], ..., E[2^{\log(q)}m])$
- Multiplication by $c \in \mathbb{Z}_q$:
  - Write $c = \sum_i c_i 2^i$, where $c_i \in \{0, 1\}$
  - Compute $\sum_i c_i E[2^i m] = E[\sum_i c_i 2^i m] = E[cm]$
- $cE'[m] = E[cm]$
- We can also compute $E'[cm]$:
  \[
  c^*E'[m] = (cE'[m], (2c)E'[m], ..., (2^{\log_q c})E'[m]) \\
  = (E[cm], E[(2c)m], ..., E[(2^{\log_q c})m]) = E'[cm]
  \]
Multiplication via Homomorphic Decryption

• Idea:
  - Encryption $E(m) = (a, as + e + m)$ is linearly homomorphic
  - Decryption $D(a, b) = b - as = m + e$ is linear in $s' = (-s, 1)$
  - We can decrypt homomorphically using an encryption of $s'$

• Details
  - Given: $E(m) = (a, b)$ and $E'(s') = (E'(-s), E'(1))$
  - Compute $E(m) * E'(s') = a * E'(-s) + b * E'(1) = E(m)$

• More interesting:
  - Given $E(m)$ and $E'(cs')$
  - Compute $E(m) * E'(cs') = E(cm)$
Homomorphic “decrypt and multiply”

- $E''(c) = E'(cs') = E'("E(m)\rightarrow c*m")$
- $E''(c) = \{E(\alpha_i c)\}_i$ for some $\alpha_i(s)$
- Homomorphiofic Properties:
  - $E''(m_1) + E''(m_2) = E''(m_1 + m_2)$
  - $E''(m_1)\cdot E''(m_2)$
    $= \{E(\alpha_i m_1)\cdot E''(m_2)\}_i$
    $= \{E(\alpha_i m_1 \cdot m_2)\}$
    $= E''(m_1 \cdot m_2)$
FHE

- $E''(m_0)+E''(m_1) = E''(m_0+m_1)$
- $E''(m_0)*E''(m_1) = E''(m_0*m_1+e)$

- Not quite a FHE yet:
  - $E''$ can evaluate any arithmetic circuit
  - But noise grows with computation

- Effectively:
  - can only evaluate small circuits / branching programs

- Bootstrapping: FHE(NC1) $\rightarrow$ FHE(PTIME)
Bootstrapping FHE

- Let $c = \text{Enc}_s(m^*(q/2)+e)$
- $f_c(s) = \text{msb(Dec}_s(c))^\ast(q/2) = m^\ast(q/2)$
- Eval $f_c$ homomorphically on $\{s\} = \text{Enc}_s(s)$
- $f_c(\{s\}) = \{f_c(s)\} = \{\text{msb(Dec}_s(c))\}$
  $\quad = \{m^\ast(q/2)\} = \text{Enc}_s(m^\ast(q/2))$
- Output noise depends on $	ext{msb}^\ast\text{Dec}_s\{s\}$, but not on $e$
Composing FHE computations

- Output noise depends on $\text{Dec}_s$, but not $c$.
- $\text{Enc}(m^*(q/2); \frac{q}{4}) \rightarrow \text{Enc}(m^*(q/2); \beta \ll \frac{q}{4})$
- Can compose arbitrarily many gates, while keeping noise small
Requirements

• Correctness:
  – Need “exact” decryption $\text{Dec}(\text{Enc}(m)) = m$
  – Achieved by scaling and rounding
    $\text{round}((q/2)m + e) = \text{msb}((q/2)m + e)$

• Circular security:
  – Need to encrypt $s$ under $E''_s$
  – Circular security of $E''_s(s)$
    still an open problem
  – Not needed for Leveled FHE
Summary

• Lattice (LWE) encryption $E$
  - Circular secure: $E_s(s)$
  - Linear approx. decryption $D(s)$
  - Transform $E \rightarrow E''$ (provably secure encryption)
    $E''$ can evaluate arbitrary (low depth) function

• Bootstrapping
  - Nonlinear (but still low depth) rounding function
  - Can be computed by $E''$
  - Open problem: circular security of $E''_s(s)$
Homomorphic Decrypt & Round

- Dec(s[i],[a[i],b]) = round(b-\sum a[i]s[i])
  = msb((q/4)+b-\sum a[i]s[i])

- Assume for simplicity s[i] \in \{0,1\}

- Write all numbers in binary:
  - b+(q/8)=\sum j 2^j b_j, -a[i]=\sum j 2^j a_{j[i]}, where b_j,a_{j[i]}\in\{0,1\}

- Want to compute and round
  - R = \sum j 2^j(b_j+\sum a_{j[i]}s[i])
    - Output most significant bit msb(R) = (2R/q) mod 2
Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb(∑...s[i])

- Homomorphic in s:
  - Enc(s[1]),.....,Enc(s[n]) → msb(∑)
Homomorphic Decryption

- $\text{Dec}(s[i],[a[i],b]) = \text{msb}(\sum \ldots s[i])$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>...</th>
<th>...</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>+</td>
<td>1</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>+</td>
<td>1</td>
<td>...</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Homomorphemic in s:
- Enc($s[1]$),..,Enc($s[n]$) $\rightarrow$ msb($\Sigma$)
Homomorphic Decryption

- \( \text{Dec}(s[i],[a[i],b]) = \text{msb}(\sum...s[i]) \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>...</th>
<th>...</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0</td>
<td>...</td>
<td>...</td>
<td>s[1]</td>
<td>s[1]</td>
<td>0</td>
<td>s[1]</td>
</tr>
<tr>
<td>+</td>
<td>s[2]</td>
<td>...</td>
<td>...</td>
<td>s[2]</td>
<td>0</td>
<td>s[2]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>+</td>
<td>s[n]</td>
<td>...</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>s[n]</td>
<td>s[n]</td>
</tr>
</tbody>
</table>

- Homomorphic in \( s \):
- \( \text{Enc}(s[1]),...,\text{Enc}(s[n]) \rightarrow \text{msb}(\sum) \)
Homomorphic Decryption

- \( \text{Dec}(s[i], [a[i], b]) = \text{msb}(\sum \ldots s[i]) \)
**Cryptographic Accumulator**

- \( \text{ACC}[v] \) holds values \( v \in \{0, \ldots, N=2(n+1)\} \)
- **Local Operations:**
  - Increment: \( \text{ACC}[v] \rightarrow \text{ACC}[v+1] \)
  - Half: \( \text{ACC}[v] \rightarrow \text{ACC}[v/2] \)
  - Mod2: \( \text{ACC}[v] \rightarrow \text{ACC}[v \mod 2] \)
- **Accum:** \( \text{ACC}[v], E''[s \in \{0,1\}] \rightarrow \text{ACC}[v+s] \)
- **Extract:** \( \text{ACC}[v] \rightarrow \text{Enc}(v=1) \)
ACC local operations

- [AP14] $\text{ACC}[v;\beta] = (c[0],\ldots,c[N])$ where
  - $c[v]$ = $\text{Enc}(1;\beta_v)$, $c[u]$ = $\text{Enc}(0;\beta_u)$
  - $\beta = \sum_i \beta_i$

- $f(\text{ACC}[v;\beta]) = \text{ACC}[f(v);\beta]$
  - $c'[v] = \sum \{ c[u] \mid f(u) = v \}$
  - $\sum \{} = \text{Enc}(0;0) = [0,0]$

- Increment, Half, Mod2: choose appropriate $f$

- Extract(ACC) = $c[1]$
Accumulate

- Accumulate: $\text{ACC}[v] + E''[s] \rightarrow \text{ACC}[v+s]$
- Compute $A_0 = \text{ACC}[v], A_1 = \text{ACC}[v+1]$
- Select:
  - $\text{ACC}[v+s] = A_s = A_0*(1-E''[s]) + A_1*E''[s]$
- All operations supported by our $E''$
Credits

- Most techniques used in this construction proposed independently in other works
  - Linearity of lattice cryptography [BM97],[LMPR08]
  - Multiplication gadget matrix (1,2,4,...) [Ajtai99],[BV]++
  - Approximate decryption [CKKS17] HEAAN
  - $E''$: essentially equivalent to [GSW13]
  - Accumulators [AP14]. See also [DM15],[CGGI16].

- Only new technique: bootstrapping via schoolbook addition algorithm
Concluding remarks

• Simple HE from basic building blocks
  – Regev LWE: mod-q variant of [BFKL93],[GRS08]
  – “CryptoComputing for NC1” [SYY99]

• FHE = Simple HE + Bootstrapping [G09]
  – Main efficiency bottleneck in practice
  – Main theoretical open problem: circular security

• Other applications? Yes!
  – Translate between FHE and MPC [CLOPS13],[BGG18],
  – Homomorphic Commitments [GSW13],[PS19]
  – Homomorphic Secret Sharing [BKS19]
  – Symmetric Crypto with algebraic structure [AMPR19]
References

- [BFKL93] Blum, Furst, Kearns, Lipton
- [BM97] Bellare, Micciancio
- [SYY99] Sander, Young, Yung
- [LMPR08] Lyubashevsky, Micciancio, Peikert, Rosen
- [GRS08] Gilbert, Robshaw, Seurin
- [G09] Gentry
- [BV11,14] Brakerski, Vaikuntanathan
- [CLOPS13] Choudhury, Loftus, Orsini, Patra, Smart
- [GGHRSW13] Garg, Gentry, Halevi, Raykova, Sahai, Waters
- [GKPVZ13] Goldwasser, Kalai, Popa, Vaikuntanathan, Zeldovich
- [GSW13] Gentry, Sahai, Waters
- [BLPRS13] Brakerski, Langlois, Peikert, Regev, Stehle
- [AP14] Alperin-Sherif, Peikert
- [CGGI16/17] Chilotti, Gama, Georgieva, Izabachene
- [CKKS17] Cheon, Kim, Kim, Song
- [BGG18] Boura, Gama, Georgieva
- [CCHLRRW19] Canetti, Chen, Holmgren, Lombardi, Rothblum, Rothblum, Wichs
- [PS19] Peikert, Shiehian
- [BKS19] Boyle, Kohl, Scholl
- [AMPR19] Alamati, Montgomery, Patranabis, Roy
Thank You!

Questions?