Approximate Homomorphic Encryption and Privacy Preserving Machine Learning

Jung Hee Cheon (SNU, CryptoLab)

Thanks to YongSoo Song, Kiwoo Lee, Andrey KIM
1. Homomorphic Encryption
2. HEAAN
3. Bootstrapping of HEAAN
4. Toolkit for Homomorphic Computation
Homomorphic Encryption

- **Integer-based HE scheme**

  - **RAD PH**\(^1\)
    - (Secret Key, Operation Key) = (a large prime \(p\), \(n = pq_0\))
    - Encryption: \(\text{Enc}(m) = m + pq \mod n\)
    - Decryption: \(\text{Enc}(m) \mod p = m\)
    - \(\text{Enc}(m_1) + \text{Enc}(m_2) = (m_1 + pq_1) + (m_2 + pq_2)\)
      \[= (m_1 + m_2) + p(q_1 + q_2)\]
      \[= \text{Enc}(m_1 + m_2)\]

  - **DGHV HE scheme** (on \(\mathbb{Z}_2\))\(^2\)
    - \(\text{Enc}(m \in \mathbb{Z}_{2^{100}}) = m + 2^{100}e + pq\)
    - **SECURE** against quantum computing
    - Use a polynomial ring \(R_q = \mathbb{Z}_q[x]/(xn + 1)\)

But, **INSECURE!**
Fully Homomorphic Encryption

- On Polynomials (RingLWE)
  - [Gen09] ideal lattice
  - NTRU: LTV12
  - Ring-LWE: BV11b, GHS13, BLLN13, HEAAAN etc

- On Integers (AGCD)
  - [DGHV10] FHE over the Integers. Eurocrypt 2010
  - CMNT11, CNT12, CCKLLTY13, CLT14, etc

- On Matrices (LWE)
  - [BV11a] Efficient FHE from (Standard) LWE. FOCS11
  - Bra12, BGV12, GSW13
Summary of Progress in HE

1. 2009~2012: Plausibility and Scalable for Large Circuits
   - [GH11] A single bit bootstrapping takes 30 minutes
   - [GHS12b] 120 blocks of AES-128 (30K gates) in 36 hours

2. 2012~2015: Depth-Linear Construction
   - [BGV12] Modulus/Key Switching
   - [Bra12] Scale Invariant Scheme
   - [HS14] IBM's open-source library Helib: AES evaluation in 4 minutes

3. 2015~Today: Usability
   - Various schemes with different advantages (HEAAN, TFHE)
   - Real-world tasks: Big data analysis, Machine learning
   - Competitions for Private Genome Computation (iDash, 2014~)
   - HE Standardization meetings (2017~)
Standardization: HomomorphicEncryption.org

Jul 2017 in Microsoft, Redmond

Mar 2018 in MIT

Oct 2018 in Toronto
# Homomorphic Encryption

## Best Performing HE Schemes

<table>
<thead>
<tr>
<th>Type</th>
<th>Classical HE</th>
<th>Fast Bootstrapping</th>
<th>Approximate Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext</td>
<td>Finite Field Packing</td>
<td>Binary string</td>
<td>Real/Complex numbers Packing</td>
</tr>
<tr>
<td>Operation</td>
<td>Addition, Multiplication</td>
<td>Look-up table &amp; bootstrapping</td>
<td>Fixed-point Arithmetic</td>
</tr>
<tr>
<td>Library</td>
<td>HElib (IBM), SEAL (Microsoft Research), Palisade (Duality inc.)</td>
<td>TFHE (Inpher, Gemalto, etc.)</td>
<td>HEAAN (SNU)</td>
</tr>
</tbody>
</table>
2. HEAAN: Approximate Homomorphic Encryption
Exact Multiplication

- The plaintext size is doubled after a multiplication.
Approximate Multiplication

HEAAN [CKKS17]

- Rescale after a multiplication
- Tracing # of significands
- Most data is processed approximately in Data analysis or ML
Homomorphic Encryption for Arithmetic of Approximate Numbers

Jung Hee Cheon\textsuperscript{1}, Andrey Kim\textsuperscript{1}, Miran Kim\textsuperscript{2}, and Yongsoo Song\textsuperscript{1}

\textsuperscript{1} Seoul National University, Republic of Korea
\{jhcheon, kimandrik, lucius05\}@snu.ac.kr
\textsuperscript{2} University of California, San Diego
mrkim@ucsd.edu

Abstract. We suggest a method to construct a homomorphic encryption scheme for approximate arithmetic. It supports an approximate addition and multiplication of encrypted messages, together with a new rescaling procedure for managing the magnitude of plaintext. This procedure truncates a ciphertext into a smaller modulus, which leads to rounding of plaintext. The main idea is to add a noise following significant figures which contain a main message. This noise is originally added to the plaintext for security, but considered to be a part of error occurring during approximate computations that is reduced along with plaintext by rescaling. As a result, our decryption structure outputs an approximate value of plaintext with a predetermined precision.

[CKKS, AC17] Homomorphic Encryption for Arithmetic of Approximate Numbers
Approximate Computation

- **Numerical Representation**
  - Encode $m$ into an integer $m \approx px$ for a scaling factor $p : \sqrt{2} \mapsto 1412 \approx \sqrt{2} \cdot 10^3$

- **Fixed-Point Multiplication**
  - Compute $m = m_1m_2$ and extract its significant digits $m' \approx p^{-1} \cdot m$
  - $1.234 \times 5.678 = (1234 \cdot 10^{-3}) \times (5678 \cdot 10^{-3}) = 7006652 \cdot 10^{-6} \mapsto 7007 \cdot 10^{-3} = 7.007$

- **Previous HE on LWE problem (Regev, 2005)**
  - $ct = \text{Enc}_{sk}(m)$, $\langle ct, sk \rangle = \frac{q}{t} m + e \pmod{q}$
  - Modulo $t$ plaintext vs Rounding operation
A New Message Encoding
- $ct = Enc_{sk}(m), \langle ct, sk \rangle = pm + e \pmod{q}$
- Consider $e$ as part of approximation error

Homomorphic Operations

<table>
<thead>
<tr>
<th>Input</th>
<th>$\mu_1 \approx pm_1, \mu_2 \approx pm_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>$\mu_1 + \mu_2 \approx p \cdot (m_1 + m_2)$</td>
</tr>
<tr>
<td>Multiplication</td>
<td>$\mu = \mu_1\mu_2 \approx p^2 \cdot m_1m_2$</td>
</tr>
<tr>
<td>Rounding</td>
<td>$\mu' \approx p^{-1} \cdot \mu \approx p \cdot m_1m_2$</td>
</tr>
</tbody>
</table>

Support for the (approximate) fixed-point arithmetic
- **Leveled** HE: $q = p^L$
Construction over the ring

- A single ctx can encrypt a vector of plaintext values $z = (z_1, z_2, ..., z_\ell)$
- Parallel computation in a SIMD manner $z \otimes w = (z_1 w_1, z_2 w_2, ..., z_\ell w_\ell)$
RLWE-based HEAAN

- Let $R = \mathbb{Z} [X] / (X^n + 1)$ and $R_q = (R \mod q) = \mathbb{Z}_q [X] / (X^n + 1)$
  - A ciphertext can encrypt a polynomial $m(X) \in R$
    - Note $(m_0 + m_1 X + \cdots) (m_0' + m_1' X + \cdots) = m_0 m_0' + (m_0 m_1' + m_0' m_1) X + \cdots$

- Decoding/Encoding function
  $R = \mathbb{Z} [X] / (X^n + 1) \subseteq \mathbb{R} [X] / (X^n + 1) \rightarrow \mathbb{C}^{n/2}$
  $$m(X) \mapsto z = (z_1, \ldots, z_{n/2}), \quad z_i = \mu(\zeta_i),$$
  where $X^n + 1 = (X - \zeta_1)(X - \zeta_1^{-1})(X - \zeta_2)(X - \zeta_2^{-1}) \cdots (X - \zeta_{n/2})(X - \zeta_{n/2}^{-1})$

- Example: $n = 4$, $\zeta_1 = \exp(\pi i / 4)$, $\zeta_2 = \exp(5\pi i / 4)$
  - $z = (1 - 2i, 3 + 4i) \mapsto m(X) = 2 - 2\sqrt{2} X + X^2 - \sqrt{2} X^3$
  - $\mapsto \mu(X) = 2000 - 2828X + 1000X^2 - 1414 X^3$
  - $\mu(\zeta_1) \approx 1000.15 - 1999.55 i$, $\mu(\zeta_2) \approx 2999.85 + 3999.55 i$
3. Bootstrapping of HEAAN
Bootstrapping

\[ c = Enc_k(m; r) \]

\[ K \]

\[ Enc(c) \]

\[ Enc(K) \]

\[ Enc_k(m; r') \]

\[ m \]

\[ r' \text{ small} \]

\[ \text{Input} \]
- Old Ciphertext with large noise
- Encrypted Secret Key

\[ \text{Process} \]
- Evaluate Decrypt circuit

\[ \text{Output} \]
- New ciphertext with small noise

Ciphertexts of a leveled HE have a limited lifespan

Refresh a ciphertext \( ct = Enc_{sk}(m) \) by **evaluating the decryption circuit homomorphically**

\[
\text{Dec}_{sk}(ct) = m \iff F_{ct}(sk) = m \text{ where } F_{ct}(*) = \text{Dec}_*(ct)
\]

Bootstrapping key \( BK = Enc_{sk}(sk) \)

\[
F_{ct}(BK) = F_{ct}(Enc_{sk}(sk)) = Enc_{sk}(F_{ct}(sk)) = Enc_{sk}(m)
\]

Homomorphic operations introduce errors \( \Rightarrow \) Fine

\[
F_{ct}(BK) = F_{ct}(Enc_{sk}(sk)) = Enc_{sk}(F_{ct}(sk) + e) = Enc_{sk}(m + e)
\]

How to evaluate the decryption circuit (efficiently)?

\[
\text{Dec}_{sk}(ct) = \langle ct, sk \rangle \mod q
\]
Approximate Decryption

\[ Dec_{sk}(ct) \leftrightarrow t = \langle ct, sk \rangle \leftrightarrow [t]_q = \mu, \]

\[ t = qI + \mu \text{ for some } |I| < K \]

- Naïve solution: polynomial interpolation on \([-Kq, Kq]\)
- Huge depth, complexity & inaccurate result
Approximate Decryption

\[ \text{Dec}_{sk}(ct) \leftrightarrow t = \langle ct, sk \rangle \leftrightarrow \lfloor t \rfloor_q = \mu, \]
\[ t = qI + \mu \text{ for some } |I| < K \]

- **Idea 1**: Restriction of domain \(|\mu| \ll q\)
Approximate Decryption

Idea 1: Restriction of domain $|\mu| \ll q$

Idea 2: Sine approximation $\mu \approx \frac{q}{2\pi} \sin \theta$ for $\theta = \frac{2\pi}{q} t$ (period: $q$, slope at 0 = 1)
Bootstrapping of HEAAN

- Sine Evaluation
Sine Evaluation
- Direct Taylor approximation
  • Huge depth & complexity
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1**: Low-degree approx. near 0
  
  \[ C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} \left(\frac{\theta}{2^r}\right)^{2k} \approx \cos(\theta/2^r) \]
  
  \[ S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} \left(\frac{\theta}{2^r}\right)^{2k+1} \approx \sin(\theta/2^r) \]
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1:** Low-degree approx. near 0
  - \( C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta / 2^r)^{2k} \approx \cos(\theta / 2^r) \)
  - \( S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta / 2^r)^{2k+1} \approx \sin(\theta / 2^r) \)
- **Idea 2:** Iterate by double-angle formula
  - \( C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \)
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- Idea 1: Low-degree approx. near 0
  - \[ C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta/2^r)^{2k} \approx \cos(\theta/2^r) \]
  - \[ S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta/2^r)^{2k+1} \approx \sin(\theta/2^r) \]
- Idea 2: Iterate by double-angle formula
  - \[ C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \]
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1**: Low-degree approx. near 0
  - \( C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta/2^r)^{2k} \approx \cos(\theta/2^r) \)
  - \( S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta/2^r)^{2k+1} \approx \sin(\theta/2^r) \)
- **Idea 2**: Iterate by double-angle formula
  - \( C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \)
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1**: Low-degree approx. near 0
  - \( C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta/2^r)^{2k} \approx \cos(\theta/2^r) \)
  - \( S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta/2^r)^{2k+1} \approx \sin(\theta/2^r) \)
- **Idea 2**: Iterate by double-angle formula
  - \( C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), \quad S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \)
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1**: Low-degree approx. near 0
  - \( C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta/2^r)^{2k} \approx \cos(\theta/2^r) \)
  - \( S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta/2^r)^{2k+1} \approx \sin(\theta/2^r) \)
- **Idea 2**: Iterate by double-angle formula
  - \( C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \)
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1**: Low-degree approx. near 0
  - \[ C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta/2^r)^{2k} \approx \cos(\theta/2^r) \]
  - \[ S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta/2^r)^{2k+1} \approx \sin(\theta/2^r) \]
- **Idea 2**: Iterate by double-angle formula
  - \[ C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), \quad S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \]
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1**: Low-degree approx. near 0
  - \( C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta/2^r)^{2k} \approx \cos(\theta/2^r) \)
  - \( S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta/2^r)^{2k+1} \approx \sin(\theta/2^r) \)
- **Idea 2**: Iterate by double-angle formula
  - \( C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \)
Sine Evaluation

- Direct Taylor approximation
  - Huge depth & complexity
- **Idea 1:** Low-degree approx. near 0
  - \[ C_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k)!} (\theta/2^r)^{2k} \approx \cos(\theta/2^r) \]
  - \[ S_0(\theta) = \sum_{k=0}^{d} \frac{(-1)^k}{(2k+1)!} (\theta/2^r)^{2k+1} \approx \sin(\theta/2^r) \]
- **Idea 2:** Iterate by double-angle formula
  - \[ C_{k+1}(\theta) = C_k^2(\theta) - S_k^2(\theta), S_{k+1}(\theta) = 2S_k(\theta) \cdot C_k(\theta) \]
  - Numerically stable & Linear complexity
Iteration vs Direct Computation

- $S_r(\theta)$ is obtained from $S_0(\theta)$ and $C_0(\theta)$ by $r$ iterations
  - One computation of Double-angle formula: 2 squarings + 1 addition
  - $r$ iterations take $2r$ squarings + $r$ additions
  - Degree of $S_r(\theta) \approx 2^r$

Direct Taylor Approximation

- $2^r$ multiplications to get $2^r$ degree approximation $T_{2^r}$ of sine function
Slot-Coefficient Switching

- Ring - based HEAAN
  - Homomorphic operations on plaintext slots, not on coefficients
  - We need to perform the modulo reduction on coefficients
- Pre/post computation before/after sine evaluation
  - Depth consumption: Sine evaluation
  - Complexity: Slot-Coefficient switchings (# of slots)

Performance of Bootstrapping

- Experimental Results
  - 127 + 12 = 139 s / 128 slots × 12 bits
  - 456 + 68 = 524 s / 128 slots × 24 bits

\[
\mathbb{R}[X]/(X^n + 1) \quad t(X) = q \cdot I(X) + \mu(X) \\
\mu(X) = \mu_0 + \cdots + \mu_{n-1}X^{n-1}
\]
### Speed of FHE

**1-bit, 1800s**
- 1800s → 0.05s (1bit)
- 1800s → 0.00046s (Amortized)

- Faster Homomorphic Discrete Fourier Transforms and Improved FHE Bootstrapping, eprint, 1073, 2018
- Intel Xeon CPU E5-2620 2.10GHz, 64RAM

**1bit, 0.7s**
- Batch Fully Homomorphic Encryption over the Integers, Eurocrypt 2013
- Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds, Asiacrypt 2016

**1bit, 0.052s**
- 180s, 531bit
- 320s, 16K bit

**1bit, 172s**
- 172s, 531bit

**1bit, 320s**
- 320s, 16K bit

**1bit, 0.052s**
- 100s, 250K bit

---

[CGGI16] Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds, Asiacrypt 2016.
[CHH18] Faster Homomorphic Discrete Fourier Transforms and Improved FHE Bootstrapping, eprint, 1073, 2018
## 2017 Track 3: Logistic Regression Training on Encrypted Data

### TRACK 3: BEST-PERFORMING TEAMS

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team</th>
<th>AUC</th>
<th>Encryption</th>
<th>Secure learning</th>
<th>Decryption</th>
<th>Overall time (mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SNU</td>
<td>0.6934</td>
<td>537.667 MB</td>
<td>10.250 mins</td>
<td>64.875 MB</td>
<td>10.360 mins</td>
</tr>
<tr>
<td>3</td>
<td>CEA LIST</td>
<td>0.6930</td>
<td>53.000 MB</td>
<td>2206.057 mins</td>
<td>238.255 MB</td>
<td>2207.363 mins</td>
</tr>
<tr>
<td>3</td>
<td>KU Leuven</td>
<td>0.6722</td>
<td>4904.000 MB</td>
<td>155.695 mins</td>
<td>7266.727 MB</td>
<td>160.912 mins</td>
</tr>
<tr>
<td>2</td>
<td>EPFL</td>
<td>0.6584</td>
<td>1011.750 MB</td>
<td>15.089 mins</td>
<td>1498.513 MB</td>
<td>16.739 mins</td>
</tr>
<tr>
<td>2</td>
<td>MSR</td>
<td>0.6574</td>
<td>1945.600 MB</td>
<td>385.021 mins</td>
<td>26299.344 MB</td>
<td>396.390 mins</td>
</tr>
<tr>
<td>X</td>
<td>Waseda*</td>
<td>0.7154</td>
<td>20.390 MB</td>
<td>2.077 mins</td>
<td>7635.600 MB</td>
<td>5.332 mins</td>
</tr>
<tr>
<td>X</td>
<td>Saarland</td>
<td>N/A</td>
<td>65536.000 MB</td>
<td>48.356 mins</td>
<td>29752.527 MB</td>
<td>57.344 mins</td>
</tr>
</tbody>
</table>

*Interactive mechanism, no complete guarantee on 80-bit security at “analyst” side

**Program ends with errors

Slide Courtesy of Xiaoqian Jiang (△ = too small iteration → hard to adapt for other data)
### 2018 Track 2: Secure Parallel Genome Wide Association Studies using HE

<table>
<thead>
<tr>
<th>Team</th>
<th>Submission</th>
<th>Schemes</th>
<th>End to End Performance</th>
<th>Evaluation result (F1-Score) at different cutoffs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Running time (mins)</td>
<td>Peak Memory (M)</td>
</tr>
<tr>
<td>A*FHE</td>
<td>A*FHE - 1 +</td>
<td>HEAAN</td>
<td>922.48</td>
<td>3,777</td>
</tr>
<tr>
<td></td>
<td>A*FHE - 2</td>
<td></td>
<td>1,632.97</td>
<td>4,093</td>
</tr>
<tr>
<td>Chimera</td>
<td>Version 1 +</td>
<td>TFHE &amp; HEAAN</td>
<td>201.73</td>
<td>10,375</td>
</tr>
<tr>
<td></td>
<td>Version 2</td>
<td>(Chimera)</td>
<td>215.95</td>
<td>15,166</td>
</tr>
<tr>
<td>Delft Blue</td>
<td></td>
<td>HEAAN</td>
<td>1,844.82</td>
<td>10,814</td>
</tr>
<tr>
<td>UC San Diego</td>
<td>Logistic Regr +</td>
<td>HEAAN</td>
<td>1.66</td>
<td>14,901</td>
</tr>
<tr>
<td></td>
<td>Linear Regr</td>
<td></td>
<td>0.42</td>
<td>3,387</td>
</tr>
<tr>
<td>Duality Inc</td>
<td>Logistic Regr +</td>
<td>CKKS (Aka HEAAN), pkg: PALISADE</td>
<td>3.8</td>
<td>10,230</td>
</tr>
<tr>
<td></td>
<td>Chi2 test</td>
<td></td>
<td>0.09</td>
<td>1,512</td>
</tr>
<tr>
<td>Seoul National University</td>
<td>SNU-1</td>
<td>HEAAN</td>
<td>52.49</td>
<td>15,204</td>
</tr>
<tr>
<td></td>
<td>SNU-2</td>
<td></td>
<td>52.37</td>
<td>15,177</td>
</tr>
<tr>
<td>IBM</td>
<td>IBM-Complex</td>
<td>CKKS (Aka HEAAN), pkg: HElLib</td>
<td>23.35</td>
<td>8,651</td>
</tr>
<tr>
<td></td>
<td>IBM- Real</td>
<td></td>
<td>52.65</td>
<td>15,613</td>
</tr>
</tbody>
</table>

Slide Courtesy of Xiaoqian Jiang
4. Toolkit for Homomorphic Computation
How to Pack

- **Packing Method**
  - HEAAN supports vector operations
  - How can we compute matrix operations for ciphertexts?

- **Matrix Encoding method**

  Encode

  \[
  c = \text{Enc}( \begin{bmatrix} 1 & 2 & \cdots & 16 \end{bmatrix} )
  \]
How to rotate

- **Packing Method**
  - Matrix addition: trivial
  - Matrix multiplication: non-trivial (exercise)
  - Row/column rotation?

$$\text{rot}(c, 1) = \text{Enc}( \begin{array}{cccc} 16 & 1 & \cdots & 15 \end{array} ) = \begin{array}{cccc} 16 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 18 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{array} )$$ (wrong)
How to rotate

- Packing Method
  - Solution: using masking vector

\[
\text{rot}(c, 1) \odot \text{mask} =
\begin{bmatrix}
16 & 1 & 2 & 3 \\
 4 & 5 & 6 & 7 \\
 8 & 9 & 10 & 11 \\
12 & 13 & 14 & 15
\end{bmatrix}
\begin{bmatrix}
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 & 2 & 3 \\
 0 & 5 & 6 & 7 \\
 0 & 9 & 10 & 11 \\
0 & 13 & 14 & 15
\end{bmatrix}
\]

\[
\text{rot}(c, -3) \odot \text{mask}' =
\begin{bmatrix}
4 & 5 & 6 & 7 \\
 8 & 9 & 10 & 11 \\
12 & 13 & 14 & 15 \\
16 & 1 & 2 & 3
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
= 
\begin{bmatrix}
4 & 0 & 0 & 0 \\
 8 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 \\
16 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\therefore \text{ColRot}(c) = \text{rot}(c, 1) \odot \text{mask} + \text{rot}(c, -3) \odot \text{mask}' =
\begin{bmatrix}
4 & 1 & 2 & 3 \\
 8 & 5 & 6 & 7 \\
12 & 9 & 10 & 11 \\
16 & 13 & 14 & 15
\end{bmatrix}
\]
Polynomial Approximation

- **Message Space**
  - Bit-wise HE: $\mathbb{Z}_2 = \{0,1\}$ with logical gates
    - Good at **gate operations** but slow at arithmetic.
  - Word-wise HE: $\mathbb{C}$ or $\mathbb{Z}_p$ with add. & mult.
    - Good at **poly evaluation** but hard to evaluate non-poly function.

- **Idea:** Use Polynomial Approximation for Non-Poly Functions!
  - Don’t use naïve Taylor Approx. It is **local** i.e. approx at a point
  - Minimize errors on an interval
  - Methods: Least Square, Chevyshev, Minimax
Application: Homomorphic Logistic Regression [HHCP]
- Need to evaluate sigmoid function: \( \frac{1}{1 + \exp(-x)} \)

- Real Financial Large Data (Provided by KCB)
- About 400,000 samples with 200 features.
- Target value = “credit score”

<table>
<thead>
<tr>
<th># Iterations</th>
<th>Learning Rate</th>
<th>Mini-batch</th>
<th>Block Size</th>
<th>Accuracy</th>
<th>AUROC</th>
<th>K-S value</th>
<th>Public Key</th>
<th>Encrypted Block</th>
<th>Total</th>
<th>Time / Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0.01</td>
<td>512</td>
<td></td>
<td>80%</td>
<td>0.8</td>
<td>50.84</td>
<td>2 GB</td>
<td>4.87 MB</td>
<td>1060 min</td>
<td>5.3 min</td>
</tr>
</tbody>
</table>
**Idea:** \( \text{Comp}(a, b) = \lim_{k \to \infty} \frac{a^k}{a^k + b^k} \) for \( a, b > 0 \)

1. Compute approximately (Take only the msb of the results)
2. Choose \( k \) as a power of 2
3. Use iteration algorithms for division

<table>
<thead>
<tr>
<th>Method</th>
<th>Scheme</th>
<th>(Amortized) Running time</th>
<th># of pairs</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-wise</td>
<td>HElib</td>
<td>( \approx 1 \text{ ms} )</td>
<td>1800</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TFHE</td>
<td>( \approx 1 \text{ ms} )</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ours (Word-wise)</td>
<td>HEAAN</td>
<td>0.73 ms</td>
<td>( 2^{16} )</td>
<td>(&lt; 2^{-8} )</td>
</tr>
</tbody>
</table>
Min/Max

- **General Max**

\[
\text{Max}(a_1, \ldots, at) = \lim_{k \to \infty} \frac{a_1^{k+1} + \cdots + a_t^{k+1}}{a_1^k + \cdots + a_t^k} \quad \text{for } a_i > 0
\]

- **2nd Max**

\[
\text{Max}(a_1, \ldots, at) = \lim_{k \to \infty} \frac{a_1^{k+1} + \cdots + a_t^{k+1} - \max^{k+1}}{a_1^k + \cdots + a_t^k - \max^k}
\]

- Applications: k-max, threshold counting, clustering, ...
Sorting on Encrypted Data

- **Fast Merge Sort**: $O(k \log k)$
  - Comparison-based algorithm doesn’t work on HE
  - We cannot check min-max condition

- **Sorting Network**: $O(k \log^2 k)$
  - Comparison network that always sort their inputs
  - Data-independent algorithm

- **Results**
  - 64 slots: about 12 min. (previous work: 42 min. [EGN+15])
  - 32,768 slots: about 10.5 hour (previous work: impractical)
Toward Homomorphic Machine Learning

- Basic Tools:
  - Packing, Matrix Operation, Comparison, Approximate $\text{inv/\text{sigmoid}}$,

- Decision Tree: Packing, Comparison

- Boosting: Comparison and Gradient Decent

- Deep Neural Network: Fast matrix operations + Approximate

- Convolution Neural Network: + Comparison
HEAAN: Summary

- HEAAN natively supports for the (approximate) fixed point arithmetic
- Ciphertext modulus $\log q = L \log p$ grows linearly
- Useful when evaluating analytic functions approximately:
  - Polynomial
  - Multiplicative Inverse
  - Trigonometric Functions
  - Exponential Function (Logistic Function, Sigmoid Function)
- Packing technique based on DFT
  - SIMD (Single Instruction Multiple Data) operation
  - Rotation on plaintext slots:
    $$ z = (z_1, ..., z_{n/2}) \mapsto z' = (z_2, ..., z_{n/2}, z_1) $$
Thank you for your attention!
감사합니다.