Homomorphic Matrix Computation & Application to Neural Networks

**Xiaqian Jiang, Miran Kim** (University of Texas, Health Science Center at Houston)

**Kristin Lauter** (Microsoft Research), **Yongsoo Song** (University of California, San Diego)
Background
Primitives for Secure Computation

- Differential Privacy
  - Limited Applications (e.g. count, average). Privacy budget.

- (Secure) Multi-Party Computation
  - Lower complexity, but higher communication costs.
  - e.g. 40GB for GWAS analysis for 100K individuals [Nature Biotechnology'17]
  - Protocol has many rounds.

- (Fully) Homomorphic Encryption
  - Higher complexity, but less communication costs.
  - One round protocol.
<table>
<thead>
<tr>
<th></th>
<th>Homomorphic Encryption</th>
<th>Multi-Party Computation</th>
</tr>
</thead>
</table>
| Re-usability | High (non-interactive)  
One-time encryption  
No further interaction from the data owners | Single-use encryption  
Not good for long-term storage  
Interaction between parties each time |
| Sources  | Unlimited                                      | Limited participants (due to complexity constraints)         |
| Speed    | Slow in computation (but can speed-up using SIMD) | Slow in communication (due to large circuit to be exchanged) |

HE is ideal for long term storage and non-interactive computation
Summary of Progresses

- **2009-10: Plausibility**
  - [GH’11] A single bit operation takes 30 minutes.

- **2011-12: Real Circuits**
  - [GHS’12] A 30,000-gate in 36 hours

- **2013-16: Usability**
  - HElib [HS’14]: IBM's open-source implementation of the BGV scheme
    - The same 30,000-gate in 4-15 minutes

- **2017-Today: Practical uses for real-world applications**
  - HE Standardization workshops
  - iDASH Privacy & Security competition (2013~)
Secure Health Data Analysis

- Predicting Heart Attack
  - ~0.2 seconds.

- Sequence matching
  - ~27 seconds, Edit distance of length 8.
  - ~180 seconds, Approximate edit distance of length 10K (iDASH'15)

- Searching of Biomarkers
  - ~0.2 seconds, 100K database (iDASH'16)

- Training Logistic Regression Model
  - ~7 minutes, 18 features * 1600 samples (iDASH'17)
Homomorphic Matrix Operation

- HElib (Crypto'14)
  - (Matrix) * (Vector)
- CryptoNets (ICML'16)
  - (Plain matrix) * (Element-wisely encrypted vector)
- GAZELLE (Usenix Security'18)
  - (Column-wisely encrypted matrix) * (Plain vector)
- Homomorphic Evaluation of (Deep) Neural Networks
  - [BMMP17] Evaluation of discretized DNN, [CWM+17] Classification on DNN.
  - Evaluation of Plain model on Encrypted data.
Homomorphic Matrix Operation

- **HElib (Crypto'14)**
  - (Matrix) * (Vector)
  - $O(d)$ complexity for (matrix*vector).
  - $\rightarrow O(d^2)$ for (matrix*matrix): not optimal.

- **CryptoNets (ICML'16)**
  - (Plain matrix) * (Element-wisely encrypted vector)

- **GAZELLE (Usenix Security'18)**
  - (Column-wisely encrypted matrix) * (Plain vector)

- **Homomorphic Evaluation of (Deep) Neural Networks**
  - [BMMP17] Evaluation of discretized DNN, [CWM+17] Classification on DNN.
  - Evaluation of Plain model on Encrypted data.
Motivation

- **Scenarios (Data/Model owner; Cloud server; Individuals)**
  1. Data owner trains a model and makes it available on the cloud.
  2. Model provider encrypts a trained model & uploads it to the cloud to make predictions on encrypted inputs from individuals.
  3. Cloud trains a model on encrypted data and uses it to make predictions on new encrypted inputs.

- **Our Work: Homomorphic Operations between Encrypted Matrices**
Main Idea
Functionality of HE Schemes

- Packing Method
  - Vector encryption & Parallel operations.
  - $\text{Enc}(x_1,..., x_n) \times \text{Enc}(y_1,..., y_n) = \text{Enc}(x_1 \times y_1,..., x_n \times y_n)$
Functionality of HE Schemes

- Packing Method
  - Vector encryption & Parallel operations.
  - \( \text{Enc}(x_1, \ldots, x_n) \ast \text{Enc}(y_1, \ldots, y_n) = \text{Enc}(x_1 \ast y_1, \ldots, x_n \ast y_n) \)

- Scalar Multiplication
  - \( (a_1, \ldots, a_n) \ast \text{Enc}(x_1, \ldots, x_n) = \text{Enc}(a_1 x_1, \ldots, a_n x_n) \)

- Rotation
  - \( \text{Enc}(x_1, \ldots, x_n) \rightarrow \text{Enc}(x_2, \ldots, x_n, x_1) \)
Functionality of HE Schemes

- Packing Method
  - Vector encryption & Parallel operations.
  - \( \text{Enc}(x_1, ..., x_n) \times \text{Enc}(y_1, ..., y_n) = \text{Enc}(x_1 \times y_1, ..., x_n \times y_n) \)

- Scalar Multiplication
  - \( (a_1, ..., a_n) \times \text{Enc}(x_1, ..., x_n) = \text{Enc}(a_1 x_1, ..., a_n x_n) \)

- Rotation
  - \( \text{Enc}(x_1, ..., x_n) \rightarrow \text{Enc}(x_2, ..., x_n, x_1) \)

- Composition of basic operations
  - Permutation, linear transformation (Expensive)

How to Represent Matrix Arithmetic Using HE-Friendly Operations?
Matrix Encoding

- Identify $n=d^2$ dimensional vector to $d \times d$ matrix
  - Addition is easy.
Matrix Encoding

- Identify $n=d^2$ dimensional vector to $d^*d$ matrix
  - Addition is easy.
  - Row or Column shifting permutations are cheap.
    (Depth 1, Complexity $O(1)$)
Matrix Encoding

- Identify $n=d^2$ dimensional vector to $d \times d$ matrix
  - Addition is easy.
  - Row or Column shifting permutations are cheap.
    (Depth 1, Complexity $O(1)$)
Matrix Multiplication

\[ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad B = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \]

\[ AB = A_0 \odot B_0 + A_1 \odot B_1 + A_2 \odot B_2 \]

\( \odot \): Element-wise Multiplication

Matrix Mult = Generation of Ai, Bi & d-homomorphic add/mult.
Matrix Multiplication

\[ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \]

\[ B = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \]

\[ AB = A_0 \odot B_0 + A_1 \odot B_1 + A_2 \odot B_2 \]

\( \odot \): Element-wise Multiplication

Matrix Mult = Generation of \( A_i, B_i \& d \)-homomorphic add/mult.
Matrix Multiplication

\[ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad B = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \quad AB = A_0 \odot B_0 + A_1 \odot B_1 + A_2 \odot B_2 \]

\[ \odot : \text{Element-wise Multiplication} \]

Matrix Mult = Generation of \( A_i, B_i \) & d-homomorphic add/mult.
Generation of $A_i$

$A_0$ Generation: $O(d)$ homomorphic operations.

$A_i = \text{ColumnShifting}(A_0, i): O(1)$ for each.
Generation of $B_i$

$B_0$ Generation: $O(d)$ homomorphic operations.

$B_i = \text{RowShifting}(B_0, i) : O(1)$ for each.
Summary

A, B : \(d \times d\) matrices
\[AB = A_0 \odot B_0 + A_1 \odot B_1 + \ldots + A_{d-1} \odot B_{d-1}\]

Generation of \(A_0, B_0\) : General permutation - \(O(d)\).
Generation of \(A_i, B_i\) 's: Column/Row shifting from \(A_0, B_0\) - \(O(d)\).
Element-wise product and summation: \(O(d)\).

Total complexity: \(O(d)\) homomorphic operations (optimal?).
Depth: 2 (scalar mult) + 1 (homo mult).
Other Operations

- **Matrix Transposition**
  - Complexity $O(d^{0.5})$ + Depth 1.

- **Parallelization**
  - When the number of plaintext slots $> d^2$.
  - Encrypt several matrices in a single ciphertext.

- **Multiplication between Non-square Matrices**
Implementation
Experimental Results

Based on the HEAAN library for fixed-point operation \((n = 2^{13})\).
All numbers have 24-bit precision.

<table>
<thead>
<tr>
<th>Dim</th>
<th>Throughput</th>
<th>Message size</th>
<th>Expansion rate</th>
<th>Encoding+ Encryption</th>
<th>Decoding+ Decryption</th>
<th>Relative time per matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HE-MatAdd</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.47 KB</td>
<td>3670</td>
<td>34 ms</td>
<td>9 ms</td>
<td>0.62 ms</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.75 KB</td>
<td>229</td>
<td>41 ms</td>
<td>12 ms</td>
<td>0.05 ms</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>12.0 KB</td>
<td>14.3</td>
<td>95 ms</td>
<td>81 ms</td>
<td>0.03 ms</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0.75 KB</td>
<td>229</td>
<td>33 ms</td>
<td>13 ms</td>
<td>0.62 ms</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>3.0 KB</td>
<td>57.3</td>
<td>48 ms</td>
<td>27 ms</td>
<td>0.19 ms</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>12.0 KB</td>
<td>14.3</td>
<td>97 ms</td>
<td>78 ms</td>
<td>0.04 ms</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>12.0 KB</td>
<td>14.3</td>
<td>108 ms</td>
<td>76 ms</td>
<td>0.62 ms</td>
</tr>
</tbody>
</table>
## Evaluation of Neural Networks

<table>
<thead>
<tr>
<th></th>
<th>Stage</th>
<th>Latency (s)</th>
<th>Relative time per image (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data owner</td>
<td>Encoding + Encryption</td>
<td>1.56</td>
<td>24.42</td>
</tr>
<tr>
<td>Model provider</td>
<td>Encoding + Encryption</td>
<td>12.33</td>
<td>-</td>
</tr>
<tr>
<td>Cloud</td>
<td>Convolution</td>
<td>5.68</td>
<td>88.75</td>
</tr>
<tr>
<td></td>
<td>1&lt;sup&gt;st&lt;/sup&gt; square</td>
<td>0.10</td>
<td>1.51</td>
</tr>
<tr>
<td></td>
<td>FC-1</td>
<td><strong>20.79</strong></td>
<td><strong>324.85</strong></td>
</tr>
<tr>
<td></td>
<td>2&lt;sup&gt;nd&lt;/sup&gt; square</td>
<td>0.06</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>FC-2</td>
<td>1.97</td>
<td>30.70</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td><strong>28.59</strong></td>
<td><strong>446.77</strong></td>
</tr>
<tr>
<td>Authority</td>
<td>Decoding + Decryption</td>
<td>0.07</td>
<td>1.14</td>
</tr>
</tbody>
</table>

1 Convolution layer + 2 Fully connected layers.
Parallel evaluation on 64 images.
## Comparison?

<table>
<thead>
<tr>
<th>Framework</th>
<th>Method</th>
<th>Runtime (s)</th>
<th>Communication (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Offline</td>
<td>Online</td>
</tr>
<tr>
<td>CryptoNets</td>
<td>HE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MiniONN</td>
<td>HE, MPC</td>
<td>0.88</td>
<td>0.40</td>
</tr>
<tr>
<td>GAZELLE</td>
<td>HE, MPC</td>
<td>0</td>
<td>0.03</td>
</tr>
<tr>
<td>E2DM</td>
<td>HE</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Plain model (previous work) vs. Encrypted model (ours)
Questions?
Thanks for listening