HybridAlpha: An Efficient Approach for Privacy-Preserving Federated Learning

Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, Heiko Ludwig
Federated Learning - Overview

How it works:
- Parties (P) collaboratively train a ML model, keeping training data to themselves
- Models are trained locally, within each party
- Local models’ parameters from parties are merged and distributed to parties, at the end or after each epoch
- Different topologies used in different trust models, often using an Aggregator (A)

Privacy Issues of Federated Learning
- privacy leakage of model output
- privacy disclosure of aggregation computation
Hybrid Approach to Federated Learning

Overview of existing FL framework

(1) Send query request

(2) Train locally and generate model $w_1$

(3) $R_1 = E_{pk_1}(w_1 + \frac{noise}{N})$

(4) Response $R_1$

(2) Train locally and generate model $w_2$

(4) Composition of $\{R_t\}$ and Aggregation $w_g = D_{sk}(\{R_t\})$

Privacy Issues of Federated Learning
- privacy leakage of model output
- privacy disclosure of aggregation computation

Current approaches
- combines secure multi-party computation (SMC) and differential privacy (DP) through reduced noise

Limitation of current approaches
- very slow due to encryption algorithm used
- require multiple rounds of communication
- do not support dynamic participation: dropouts and new additions are not allowed without full system re-keying
- cannot prevent curious aggregators from getting partial decrypted data
## Comparison of privacy-preserving approaches in federated learning framework

<table>
<thead>
<tr>
<th>Proposed Approach</th>
<th>Threat Model</th>
<th>Privacy Guarantee</th>
<th>SMC</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>participant</td>
<td>computation</td>
<td>type</td>
<td>communication</td>
</tr>
<tr>
<td></td>
<td>aggregator</td>
<td>output</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Shokri and Shmatikov [36]</td>
<td>honest</td>
<td>✓</td>
<td>-</td>
<td>1 round</td>
</tr>
<tr>
<td>PATE [31]</td>
<td>honest</td>
<td>✓</td>
<td>-</td>
<td>1 round</td>
</tr>
<tr>
<td>PySyft [34]</td>
<td>honest</td>
<td>✓</td>
<td>HE</td>
<td>2 rounds ‡</td>
</tr>
<tr>
<td>Bonawitz et al. [6]</td>
<td>dishonest</td>
<td>✓</td>
<td>SS+AE</td>
<td>3 rounds ‡</td>
</tr>
<tr>
<td>Truex et al. [38]</td>
<td>dishonest</td>
<td>✓</td>
<td>TP</td>
<td>3 rounds ‡</td>
</tr>
<tr>
<td><strong>HybridAlpha (our work)</strong></td>
<td>dishonest</td>
<td>✓</td>
<td>FE</td>
<td>1 round ‡</td>
</tr>
</tbody>
</table>

© 2019 IBM Corporation
Comparison of SMC-based Secure Aggregation

Steps of SMC aggregation
1. key setup & local training
2. encrypt model parameters $c_{\text{init}}$
3. send $c_{\text{init}}$ to $\mathcal{A}$
4. combine cipher $c_{\text{cmb}} \leftarrow \{c_{\text{init}}\}$
5. send back $c_{\text{cmb}}$
6. partial decrypt $c_{\text{part}} \leftarrow c_{\text{cmb}}$
7. send $c_{\text{part}}$ to $\mathcal{A}$
8. (combine) decryption
9. update global model

Issues
- Current SMC protocols are not efficient enough
  - crypto efficiency (time)
  - communication steps
- Lack of support for dynamic participants

Table 2: The number of crypto-related operations required for each solution.

<table>
<thead>
<tr>
<th>Communication</th>
<th>TP-SMC</th>
<th>P-SMC</th>
<th>HybridAlpha *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step (1)</td>
<td>$n$</td>
<td>$n$</td>
<td>$n + m$</td>
</tr>
<tr>
<td>Step (3)</td>
<td>$n \times m$</td>
<td>$n \times m$</td>
<td>$n \times m$</td>
</tr>
<tr>
<td>Step (5)</td>
<td>$m \times t$</td>
<td>$n \times m$</td>
<td>-</td>
</tr>
<tr>
<td>Step (7)</td>
<td>$t \times m$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TOTAL</td>
<td>$2mt + mn + n$</td>
<td>$2mn + n$</td>
<td>$mn + m + n$</td>
</tr>
</tbody>
</table>
Functional Encryption in a Nutshell

• In functional encryption for inner-product, a third-party authority that
  - generates the public key $pk$ for encryptor to encrypt vector $x = [x_1, \ldots, x_n]$
  - generates functional private key $sk_{f,y}$ that is corresponding to a vector $y$ for the decryptor

• From now on, let us assume, there is a trusted third party other than aggregator to be involved in the federated learning

**Functional Encryption for Inner-product [\*]**

$$f(x, y) = \sum x_i y_i$$

$$D_{sk} \left( f \left( E_{pk} (x_1, \ldots, x_n), y \right) \right) = \sum x_i y_i$$

without learning $x_1, \ldots, x_n$

$x_1, \ldots, x_n$ can be from one single source or multiple sources

Non-interactive Secure Computation

secure multi-party aggregation computation

- Parties $P = (p_1, ..., p_i, ..., p_n)$, each party $p_i$ has input $x_i$
- Aggregator acquire $\sum(x_i)$ without learning specific $x_i$ of $p_i$

(1) Initialization: acquires public key $\{pk_i\}$ for each participant

(2) $p_i$ prepares inputs $x_i$ and encrypt $\{c_i = E_{MIFE,pk_i}(x_i)\}$, respectively

(3) Send $\{c_i\}$

(Non-interactive communication)

(4) Collect $\{c_i = E_{MIFE,pk_i}(x_i)\}$

(5) Decrypt $\nu = D_{MIFE,sk_f}(\{c_i\})$, where $\nu$ is the result of $\sum(x_i)$

Constructed from Multiple-input Functional Encryption (MIFE)*

Threat Model

• A trusted third party (TPA) that distributes keys
• An honest but curious aggregator, and the aggregator may collude with dishonest parties
• Parties may try to infer data from other participants through the final model or during the federated learning process
Overview of HybridAlpha

Hybrid Approach
- differential privacy + noise reduction through SMC
- privacy guarantee: model output / aggregation

Efficiency Improvement
- Efficient encryption/decryption algorithm
- Non-interactive secure computation

Support Dynamic Participants
- Randomly drop out / join in

(7) Inference Prevention Module

(1) Setup
msk
mpk
TPA

(2) Public Key Distribution

(3) Add DP-noise \( w_{DP} \)

(4) Encrypt and Send \( E_{pk_i}(w_{1DP}) \)

(5) Collect \( \{E_{pk_i}(W_{iDP})\} \)

(6) \( w_p = (w_{p1}, w_{p2}, \ldots, w_{pn}) \)

(8) If (7) is ok, generate secret for \( w_p \): \( Sk_{f,w_p} \)

(9) Decrypt
\( w_A = \sum(w_{p1}w_{1DP}) = D_{sk_{f,w_p}}(\{E_{pk_i}(W_{iDP})\}) \)
Inference Prevention Module

• Threshold $t$ helps detect and stop attacks from curious aggregators and colluding participants, $t$ defines a threshold on the number of non-colluding participants

• For example, if $t = 3$, the module filters the following suspicious weight vector $w_p$:
  - infers one party’s model update:
    • $<0,0,0,1>$$
    • $<0.0009,0.009,0,1>$$
    • $<1>$$
  - exclude honest parties’ model update:
    • $<1,1,0,0>$$

• $t$ has an impact on the number of dropouts allowed by the system
  - Mainly, it helps set up the minimum quorum of participants replying to the system

Algorithm 2: Inference prevention filter

| Input: $w_p$: A weighted vector to be inspected for inference attacks; $t$: threshold of minimum number of dropouts and expected number of non-colluding participants |
| function $inference$-prevention-filter($w_p$, $t$) |
| $c_{nz} \leftarrow$ count the non-zero element in $w_p$; |
| if $c_{nz} < t$ then return "invalid $w_p$"; |
| foreach non-zero $w_{p_i} \in w_p$ do |
|   if $w_{p_i} \neq \frac{1}{c_{nz}}$ then return "invalid $w_p$"; |
| end |
| forward $w_p$ to the TPA; |
Experimental Results

Cryptosystems Implementation
- Python + GMP/Charm-crypto library

Experimental Environment
- 44 core Intel Xeon E5-2699 v4 platform with 384 GB of RAM
- CNN on MNIST dataset

Baselines
- FL without DP/ FL local DP
- TP-SMC FL (DP)
- P-SMC FL (DP)

On average reduces the training time by 68%
the data transfer volume by 92%
While providing the same model performance
the same privacy guarantees as the existing solutions
Thank you! Questions?

Find our AI Security and Privacy Solutions team at:
https://resedit.watson.ibm.com/researcher/view_group.php?id=10276
References


