Outline

● Motivation + Problem Statement
● Background + Threat Model
● Overview of Helen + Key Features
● Results
● Discussion
Outline

● Motivation + Problem Statement
● Background + Threat Model
● Overview of Helen + Key Features
● Results
● Discussion
Motivation #1
Motivation #2
Problem

“collaboratively train machine learning models on combined datasets for a common benefit”

“organizations cannot share their sensitive data in plaintext due to privacy policies and regulations or due to business competition”
Outline

● Motivation + Problem

● Background + Threat Model

● Overview of Helen + Key Features

● Results

● Discussion
Background on Coopetitive Learning

- Coopetitive -> cooperative and competitive
- Secure multi-party computation (MPC)
  - inefficient
- Previous works are limited
  - unrealistic threat models
  - limited to two parties

Fig. 1: The setting of coopetitive learning.
Threat Model

- malicious setting - only trust yourself!
- all other parties can misbehave/be malicious during protocol
- all parties agree on a functionality to compute
- confidentiality of final model not protected
Background on Crypto Building Blocks

- threshold partially homomorphic encryption
  - partially homomorphic
    - ex. Paillier \( \rightarrow \) \( \text{Enc}(X) \times \text{Enc}(Y) = \text{Enc}(X+Y) \)
  - threshold
    - need enough shares of secret key to decrypt
- zero knowledge proofs
  - prove that a certain statement is true without revealing the prover’s secret
- secure multi party computation
  - jointly compute a function over inputs while keeping inputs private
  - SPDZ chosen over garbled circuits because matrix operations are more efficient
Overview of Helen

- platform for maliciously secure coopetitive learning
- supports regularized linear models
  - paper notes that these types of models are widely used
- few organizations, lots of data, smaller number of features
Key Features of Helen

- Overarching goal: Make expensive cryptographic computation independent of number of training samples
- Make all parties commit to input dataset and prove it
- Use ADMM (Alternating Direction Method of Multipliers)/LASSO
- Use partially homomorphic encryption to encrypt global weights such that each party can compute in a decentralized manner
- 5 phases
  - Agreement Phase
  - Initialization Phase
  - Input Preparation Phase
  - Model Compute Phase
  - Model Release Phase
Input Preparation Phase

- **Goal:** broadcast encrypted summaries of data and commit
- **Why?** Malicious parties could use inconsistent data during protocol
- **How?** Encrypt data and attach various proofs of knowledge
- **Naive method:** commit on input dataset
  - crypto computation scales linearly
  - requires complex matrix inversions in MPC

\[
A_i = (X_i^T X_i + \rho I)^{-1}
\]

\[
b_i = X_i^T y_i
\]
Input Preparation Phase

- Goal: broadcast encrypted summaries of data and commit
- Why? Malicious parties could use inconsistent data during protocol
- How? Encrypt data and attach various proofs of knowledge
- **Better method: Decompose A and b via SVD**
  - all of these matrices are dimension $d$, no longer $n$
  - Each party broadcasts encrypted $A$, $b$, $y^*$, $V$, $Σ$, $Θ$ along with proofs of knowledge

\[
A_i = (X_i^T X_i + ρI)^{-1}
\]

\[
b_i = X_i^T y_i
\]

\[
A = VΘV^T,
\]

\[
b = VΣ^Ty^*,
\]
Input Preparation Phase

- End of input preparation phase
Model Compute Phase

- Goal: run ADMM algorithm iteratively and update encrypted global weights
- Why ADMM?
  - efficient for linear models
  - converges in few iterations (10)
  - supports decentralized computation
  - reduces number of expensive MPC syncs
  - thus, efficient for cryptographic training
Model Compute Phase

- Goal: run ADMM iteratively to update encrypted global weights

- 1. Local optimization
  - Each party calculates $\text{Enc}(w_i^{k+1})$
  - Also generate a proof of this

- 2. Coordination using MPC
  - Parties use input summaries to verify $\text{Enc}(w_i^{k+1})$
  - Convert weights to MPC
  - Compute softmax via MPC
  - Convert $z$ back into encrypted form

---

The competitive learning task for LASSO

Input of party $P_i$: $X_i, y_i$

1) $A_i \leftarrow (X_i^T X_i + \rho I)^{-1}$
2) $b_i \leftarrow X_i^T y_i$
3) $u^0, z^0, w^0 \leftarrow 0$
4) For $k = 0$, ADMMIterations-1:
   a) $w_i^{k+1} \leftarrow A_i(b_i + \rho(z^k - u_i^k))$
   b) $z^{k+1} \leftarrow S_{\lambda/m\rho}(\frac{1}{m} \sum_{i=1}^{m} (w_i^{k+1} + u_i^k))$
   c) $u_i^{k+1} \leftarrow u_i^k + w_i^{k+1} - z^{k+1}$
Model Release Phase

- Goal: jointly decrypt and release model parameters (z)
  - ciphertext to MPC conversion
  - verify this conversion
  - jointly decrypt model parameters (z)
Outline

● Motivation + Problem
● Background + Threat Model
● Overview of Helen + Key Features
● Results
● Discussion
Results

• Evaluation of runtime of Helen’s different phases using a synthetic dataset
## Results

<table>
<thead>
<tr>
<th>Samples per party</th>
<th>2000</th>
<th>4000</th>
<th>6000</th>
<th>8000</th>
<th>10K</th>
<th>40K</th>
<th>100K</th>
<th>200K</th>
<th>400K</th>
<th>800K</th>
<th>1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>sklearn L2 error</td>
<td>8937.01</td>
<td>8928.32</td>
<td>8933.64</td>
<td>8932.97</td>
<td>8929.10</td>
<td>8974.15</td>
<td>8981.24</td>
<td>8984.64</td>
<td>8982.88</td>
<td>8981.11</td>
<td>8980.35</td>
</tr>
<tr>
<td>Helen L2 error</td>
<td>8841.33</td>
<td>8839.96</td>
<td>8828.18</td>
<td>8839.56</td>
<td>8837.59</td>
<td>8844.31</td>
<td>8876.00</td>
<td>8901.84</td>
<td>8907.38</td>
<td>8904.11</td>
<td>8900.37</td>
</tr>
<tr>
<td>sklearn MAE</td>
<td>57.89</td>
<td>58.07</td>
<td>58.04</td>
<td>58.10</td>
<td>58.05</td>
<td>58.34</td>
<td>58.48</td>
<td>58.55</td>
<td>58.58</td>
<td>58.56</td>
<td>58.57</td>
</tr>
<tr>
<td>Helen MAE</td>
<td>57.23</td>
<td>57.44</td>
<td>57.46</td>
<td>57.44</td>
<td>57.47</td>
<td>57.63</td>
<td>58.25</td>
<td>58.38</td>
<td>58.36</td>
<td>58.37</td>
<td>58.40</td>
</tr>
</tbody>
</table>

**TABLE II**: Select errors for gas sensor (due to space), comparing Helen with a baseline that uses sklearn to train on all plaintext data. L2 error is the squared norm; MAE is the mean average error. Errors are calculated after post-processing.

<table>
<thead>
<tr>
<th>Samples per party</th>
<th>1000</th>
<th>2000</th>
<th>4000</th>
<th>6000</th>
<th>8000</th>
<th>10K</th>
<th>20K</th>
<th>40K</th>
<th>60K</th>
<th>80K</th>
<th>100K</th>
</tr>
</thead>
<tbody>
<tr>
<td>sklearn L2 error</td>
<td>92.43</td>
<td>91.67</td>
<td>90.98</td>
<td>90.9</td>
<td>90.76</td>
<td>90.72</td>
<td>90.63</td>
<td>90.57</td>
<td>90.55</td>
<td>90.56</td>
<td>90.55</td>
</tr>
<tr>
<td>Helen L2 error</td>
<td>93.68</td>
<td>91.8</td>
<td>91.01</td>
<td>90.91</td>
<td>90.72</td>
<td>90.73</td>
<td>90.67</td>
<td>90.57</td>
<td>90.54</td>
<td>90.57</td>
<td>90.55</td>
</tr>
<tr>
<td>Helen MAE</td>
<td>6.92</td>
<td>6.82</td>
<td>6.77</td>
<td>6.78</td>
<td>6.79</td>
<td>6.81</td>
<td>6.80</td>
<td>6.79</td>
<td>6.80</td>
<td>6.80</td>
<td>6.80</td>
</tr>
</tbody>
</table>

**TABLE III**: Errors for song prediction, comparing Helen with a baseline that uses sklearn to train on all plaintext data. L2 error is the squared norm; MAE is the mean average error. Errors are calculated after post-processing.
Results
Outline

● Motivation + Problem
● Background + Threat Model
● Overview of Helen + Key Features
● Results
● Discussion
Discussion

- Is there a need to extend to other types of models? Consequences of this?
- Trusted hardware (enclaves) is another popular approach to computing on sensitive data. Is it more viable?
- What happens when more parties get involved? Comparison vs. federated learning?
- Questions?