Privacy-Preserving Collaborative Filtering with SPDZ

Master Thesis Presentation by Thibaud Kehler, 321038
Thesis Scope

Privacy-Preserving
- Unlinkability of…
- Anonymity of…
- Confidentiality of…

…personal informations.

Collaborative Filtering
- Predict ratings,
  - with data-mining algorithms
  - based on personal ratings.

with SPDZ
- Multi-party computation
- Open-source implementation
- Strong security guarantees

Feasibility Study: Is SPDZ a practical approach?
- Define requirements for privacy-preserving collaborative filtering
- Explore the possibilities of SPDZ
- Implement a prototype with minimum features
- Evaluate w.r.t. the maturity of SPDZ, accuracy and performance
Outline

Thesis Scope
Use Cases and Requirements
  Use Case
  Threat Analysis
  Requirements
Preliminaries
  Secure Multi-Party Computation
  SPDZ Protocol
  Neighbourhood-Based Collaborative Filtering

Collaborative Filtering with SPDZ
  Rating Matrix
  Similarity Model
  Prediction
  Summary of our approach
  Implementation
Evaluation
  Accuracy
  Performance
  Privacy and Robustness
Conclusion
  Future Work and Open Questions
Use Case

Medical Data

Patient data is confidential by law!
Threat Analysis

**Items of Interest (IOI)**
- personal information
- recommendations

**Linkability** Are two IOIs linked?

**Identifyability** Who is the subject of an IOI?

**Disclosure of Information** IOI is disclosed to some unauthorized entity.
Requirements

Accuracy  Recommendations/Predictions are valuable to the user.

Algorithmic Complexity
- Responses in reasonable time
- Good scaling behaviour

Privacy
- Unlikability
- Anonymity
- Confidentiality

Robustness  A user of the recommender system has only minimal influence on the output.

Does collaborative filtering with SPDZ meet these requirements?
Secure Multi-Party Computation

- Parties $P_1, \ldots, P_N$
- Function $f(x_1, \ldots, x_N) = (y_1, \ldots, y_N)$
- Adversary controls the corrupted parties $C \subset \{P_1, \ldots, P_N\}, |C| = t$

Secure Multi-Party Computation (Idea)$^{[3]}$
Together the parties evaluate the function $f$ while guaranteeing the following properties
- $(y_1, \ldots, y_N)$ are evaluated to the correct values (correctness)
- $y_i$ is the only new information that is revealed to $P_i$ (privacy)

Definition (Protocol Privacy)$^{[3]}$
Let $\{\text{view}_i\}_{P_i \in C}$ be the leaked values and $\{x_i, y_i\}_{P_i \in C}$ be the allowed values.
Protocol is private if there exists an efficient simulator $S$ such that the simulated values $S(\{x_i, y_i\}_{P_i \in C})$ and $\{x_i, y_i\}_{P_i \in C}$ have the same random distribution.

Definition (Protocol Robustness)$^{[3]}$
A protocol is robust if there exists an efficient simulator $S$ such that for every adversary attacking the protocol, $S$ can efficiently compute an allowed influence with the same effect.
Security and Adversary Models [3]

Security

What is efficient?

Unconditional There are no constraints at all! Secure against unlimited resources.

Also: information theoretical security

Computational Adversary has limited computational resources
e.g. he only does polynomial time algorithms.

Statistical Adversary has to make extraordinarily lucky guesses.

Adversary

Constraints on adversary’s behaviour.

Active / Malicious Arbitrary and adapted behaviour.

Static / Non-Adaptive Arbitrary behaviour, but not adaptive to other parties.

Passive / Honest but curious Follows the protocol.

Adversary controls the corrupted parties $C \subset \{P_1, \ldots, P_N\}, |C| = t$

Minority $t < N/2$

Majority $N/2 \leq t \leq N - 1$
Distributed Architecture

Centralized

Fully distributed

user

ratings

collaborative filter

database

user

user

prediction

control server

SPDZ

user

SPDZ

SPDZ

user
Distributed Architecture

Federated

![Diagram showing a distributed architecture with federated user and database nodes connected through SPDZ.]
**SPDZ Protocol**

- Introduced by Damgard et al. [4]
- Operates on ⟨·⟩-shared secrets in field $\mathbb{F}_p$
- Two Phases: offline and online.
  - Offline:
    - Generic
    - Generates triples, pairs, mask values and bits.
    - Slow and much communication.
  - Online:
    - Addition and constant multiplication require no communication.
    - Multiplication and input consume pre-processing data and require one round of communication.
    - Fixed-point division, truncation, square root available but are expensive.
- Statistically secure against active (malicious) adversarial majority ($N - 1$)
- Compiler and virtual processor for Python-like programs.
Additive Secret Sharing\cite{4}

- Parties $P_1, \ldots, P_N$
- Finite field $\mathbb{F}_p$ with characteristic $p$.
- Private values are additively shared

**Definition (Additive Secret Sharing)**

The secret value $a \in \mathbb{F}_p$ is shared between the parties, if each party $P_i$ holds a *uniformly random* value $a_i \in \mathbb{F}_p$ such that

$$a = a_1 + \cdots + a_N$$

**Example**

- Parties $P_1, P_2$
- Finite field $\mathbb{F}_{11} = \{-5, -4 \ldots, 0, \ldots, 4, 5\}$

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>-2</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
Preprocessing/Offline Phase

- Generic Pre-Processing Phase (Offline Phase)
- Based on somewhat fully homomorphic encryption.
- Generates enough
  - Multiplication Triples
    \[ \langle a \rangle \cdot \langle b \rangle = \langle c \rangle \]
  - Square Pairs
    \[ \langle a \rangle^2 = \langle b \rangle \]
  - Random Bits
    \[ \langle b \rangle \text{ with } b \in \{0, 1\} \]
  - Input Mask Values
    \[ \langle r \rangle \text{ with } r \in \mathbb{F}_p \]
- Values are uniformly random and shared but unknown to all parties.
- Complexity \( \mathcal{O}(N^3) \)
Privacy Preserving Recommender Systems with SPDZ
Thibaud Kehler | 321038
Informatik 5 | Information Systems | RWTH Aachen
27.06.2018
Fixed-Point Numbers

- Precision $f$
- Fixed-point number $x$ is converted to the integer
  \[ v_x = x \cdot 2^f \]
- Multiplication $x \cdot y$ requires Truncation:
  \[ \frac{v_x \cdot v_y}{2^f} = \frac{x \cdot y \cdot 2^{2f}}{2^f} = x \cdot y \cdot 2^f = v_{xy} \]
- Truncation and division algorithms [2] already implemented in SPDZ.
- Square-root algorithms [6] were missing.
- We write $\langle x \rangle_f$ for the fixed-point representation of precision $f$. 
Neighbourhood-Based Collaborative Filtering

- Data mining algorithm
- Founded on the neighbourhood-based classification problem
  - User-based model or
  - Item-based model

**Definition (Collaborative Filter (Idea)[1])**

Given a set of users $U$ and items $I$ as well as an incomplete set of ratings $R$, based on these ratings predict the rating $r_{ui}$ a user $u$ would probably give to an item $i$.

- Similarity model, e.g. cosine similarity

\[
\text{Cosine}(u, v) := \frac{\sum_{i \in I_u \cap I_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_u \cap I_v} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} r_{vi}^2}}.
\]

- Prediction

\[
\hat{r}_{ui} = \frac{\sum_{v \in P_u} \text{sim}(u, v) \cdot r_{vi}}{\sum_{v \in P_u} |\text{sim}(u, v)|}.
\]
Neighbourhood-Based Collaborative Filtering

- Similarity model for user 1.

<table>
<thead>
<tr>
<th>user v</th>
<th>item i</th>
<th>Cosine(u = 1, v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>∅ 1 3 4 ∅</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>2 1 ∅ 4 ∅</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>∅ 2 3 ∅ 3</td>
<td>0.96</td>
</tr>
<tr>
<td>4</td>
<td>5 ∅ 2 1 ∅</td>
<td>0.89</td>
</tr>
<tr>
<td>5</td>
<td>4 3 ∅ 1 4</td>
<td>0.53</td>
</tr>
</tbody>
</table>

- Predict rating from user 1 for item 5.

\[
\hat{r}_{15} = \frac{0.96 \cdot 3}{0.96} = 3.
\]
Collaborative Filtering with SPDZ
Collaborative Filtering with SPDZ

User-Based Model
1. Enter the rating matrix as private input
   – in plain representation.
   – in sparse representation.
2. Build a user-based cosine similarity model.
3. Predict the ratings securely.
4. Reveal predictions only to their users.

Item-Based Model
1. Enter the rating matrix as private input
   – in plain representation.
2. Build an item-based cosine similarity model.
3. Reveal the similarity model.
4. Users predict their ratings locally.
Rating Matrix

Plain Representation

\[
R = \begin{bmatrix}
\langle r_{11} \rangle_f & \cdots & \langle r_{1m} \rangle_f \\
\vdots & \ddots & \vdots \\
\langle r_{n1} \rangle_f & \cdots & \langle r_{nm} \rangle_f 
\end{bmatrix} \quad R2 = \begin{bmatrix}
\langle r^2_{11} \rangle_f & \cdots & \langle r^2_{1m} \rangle_f \\
\vdots & \ddots & \vdots \\
\langle r^2_{n1} \rangle_f & \cdots & \langle r^2_{nm} \rangle_f 
\end{bmatrix} \quad B = \begin{bmatrix}
\langle b_{11} \rangle & \cdots & \langle b_{1m} \rangle \\
\vdots & \ddots & \vdots \\
\langle b_{n1} \rangle & \cdots & \langle b_{nm} \rangle 
\end{bmatrix}
\]

- \( \emptyset^2 := \emptyset \)
- \( \langle \emptyset \rangle_f = \langle 0 \rangle_f \)
- \( b_{ui} \in \{0, 1\} \) denotes if user \( u \) has rated item \( i \).
Similarity Model

\[ R = \begin{bmatrix} \langle r_{11} \rangle_f & \cdots & \langle r_{1m} \rangle_f \\ \vdots & \ddots & \vdots \\ \langle r_{n1} \rangle_f & \cdots & \langle r_{nm} \rangle_f \end{bmatrix} \quad R2 = \begin{bmatrix} \langle r_{11}^2 \rangle_f & \cdots & \langle r_{1m}^2 \rangle_f \\ \vdots & \ddots & \vdots \\ \langle r_{n1}^2 \rangle_f & \cdots & \langle r_{nm}^2 \rangle_f \end{bmatrix} \quad B = \begin{bmatrix} \langle b_{11} \rangle & \cdots & \langle b_{1m} \rangle \\ \vdots & \ddots & \vdots \\ \langle b_{n1} \rangle & \cdots & \langle b_{nm} \rangle \end{bmatrix} \]

Cosine Similarity

\[
\text{Cosine}(u, v) := \frac{\sum_{i \in I_u \cap I_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_u \cap I_v} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} r_{vi}^2}}
\]

\[ \langle S \rangle_{\text{user}} = \begin{bmatrix} \langle s_{11} \rangle_f & \cdots & \langle s_{1n} \rangle_f \\ \vdots & \ddots & \vdots \\ \langle s_{nn} \rangle_f \end{bmatrix} \]
Prediction

\[ R = \begin{bmatrix}
\langle r_{11} \rangle_f & \cdots & \langle r_{1m} \rangle_f \\
\vdots & \ddots & \vdots \\
\langle r_{n1} \rangle_f & \cdots & \langle r_{nm} \rangle_f
\end{bmatrix} \quad B = \begin{bmatrix}
\langle b_{11} \rangle_f & \cdots & \langle b_{1m} \rangle_f \\
\vdots & \ddots & \vdots \\
\langle b_{n1} \rangle_f & \cdots & \langle b_{nm} \rangle_f
\end{bmatrix} \quad \langle S \rangle_{\text{user}} = \begin{bmatrix}
\langle s_{11} \rangle_f & \cdots & \langle s_{1n} \rangle_f \\
\vdots & \ddots & \vdots \\
\langle s_{nn} \rangle_f
\end{bmatrix} \]

\( \varepsilon \)-Threshold Predictions

- Peer Group \( P_u(\varepsilon) = \{ v \in U_i \mid s_{uv} \geq \varepsilon \} \)

\[ \hat{r}_{ui} = \frac{\sum_{v \in P_u} s_{uv} \cdot r_{vi}}{\sum_{v \in P_u} |s_{uv}|} \]

\[ \langle \hat{r}_{ui} \rangle_f = \frac{\sum_{v \in U} \langle s_{uv} \geq \varepsilon \rangle \cdot \langle s_{uv} \rangle_f \cdot \langle r_{vi} \rangle_f}{\sum_{v \in U} \langle s_{uv} \geq \varepsilon \rangle \cdot \langle s_{uv} \rangle_f} \]

- Is there a good universal parameter \( \varepsilon \)?
Approximate $k$-nearest-neighbour predictions

$k$-Nearest Neighbour Predictions
- Better accuracy than threshold predictions.
- Peer Group $P_u(k)$ are $k$ most similar users.
- Requires many comparisons: $O(k \cdot n)$

Approximate $k$-Nearest Neighbour Predictions
- Peer Group $P_u(k)$ are approximately $k$ most similar users.
- Binary search of good parameter $\varepsilon_k$
  \[ P_u(\varepsilon_k) \approx P_u(k) \]
- Abort search after $f'$ rounds
- Use $\varepsilon_k$ for threshold prediction.
- $O(f' \cdot n)$ comparisons.
- e.g. $f' = 4$
Secure predictions are expensive. Can we compute them locally?

- For item-based predictions a needs
  1. his own ratings
  2. a similarity model
- But the similarity model could contain identifiable information!
- SPDZ cannot protect the output.

Decision: We take the risk.
Summary of our approach

User-Based Model
1. Enter the rating matrix as private input
   – in plain representation.
   – in sparse representation.
2. Build a user-based cosine similarity model.
3. Predict the ratings securely.
4. Reveal predictions only to their users.

Item-Based Model
1. Enter the rating matrix as private input
   – in plain representation.
2. Build an item-based cosine similarity model
3. Reveal the similarity model.
4. Users predict their ratings locally.

Build Model
Plain Ratings $\Theta(n^2 \cdot m)$
Sparse Ratings $\Theta(n^2 \cdot c^2)$
Approx. k-NN Predictions $\Theta(f' \cdot n)$

Item-Based Model
Plain Ratings $\Theta(m^2 \cdot n)$
k-NN Predictions Computed locally
Implementation

- SPDZ Tests
  - Programs/Source/*.mpc
- Test Suit
  - test.py
- I/O
  - io.py
- Dataset Loader
  - dataset.py
- Collaborative Filter
  - collaborative_filter.py
    - plain user-based cosine CF
    - sparse user-based cosine CF
    - item-based cosine CF
- MovieLens
- Input Files

**SPDZ Library**
- Square Root
- Sparse Types

- Python module
- MPC module
- Mixed module (Python & MPC)

**Our Contribution**
- Modified
Implementation

Baseline

Baseline Tests
test_baseline.py

Test Suit
test.py

Dataset
dataset.py

Baseline Collaborative Filter
baseline.py

MovieLens

user-based cosine CF
item-based cosine CF

Python module
Mixed module (Python & MPC)
Evaluation

- MovieLens dataset
  - 1 million ratings from 671 users for 9125 movies
  - 5-star ratings with half-star increments
  - We mean-centered the ratings user-wise
- Tested only two parties
- Used insecure but faster mock-up offline phase
Accuracy

- Random sampling $S = \{r_1, \ldots, r_s\}$
- Root Mean Square Error

$$\text{RMSE} := \sqrt{\frac{1}{s} \sum_{r \in S} (r - \hat{r})^2}$$

- Mean absolute error

$$\text{MAE} := \frac{1}{s} \sum_{r \in S} |r - \hat{r}|$$

User-Based Model

- RMSE $\approx 0.7$ and MAE $\approx 0.5$
- No major deviation from baseline.

Item-Based Model

- RMSE $\approx 0.5$ and MAE $\approx 0.4$
- No major deviation from baseline.
Performance

User-Based Model

- 3000 items
- \( \approx 78 \text{ times slower} \)
Performance

User-Based Predictions

- 3000 items
- $k = 11$, $f' = 4$
- $\approx 2560$ times slower
Performance

Item-Based Model

- 200 users
- $\approx 87$ times slower
Privacy and Robustness

Privacy

User-Based  Secure similarity model and predictions

Item-Based  Public similarity model (De-Anonymisation possible!)

Robustness

• Same robustness as idealistic third party.

• Collaborative filtering is not robust by itself!
Contribution

- Input mechanism for rating matrix.
- Prototypic implementation
- Modules for user-based and item-based collaborative filters
- Almost the same accuracy as the non-private baseline
- 2 to 3 orders of magnitude overhead for two parties.
Conclusion

Feasibility Study: Is SPDZ a practical approach?

Yes, but …

- Performance overhead
- Especially suitable for settings where privacy is important.
- Overnight batch processing.
- SPDZ itself is rather prototypic
Future Work and Open Questions

• Test our approach against a bigger data set.
• Study the possibility of model caching
• Parallelize the algorithms
• Implement other more robust recommender system algorithms
Thank you for your attention
Any questions?
References


Use Cases

2. Investment Recommendation

- Investor provides information:
  - financial interest
  - other investments
  - financial situation
- Provider could exploit this information for insider business!

3. Insurance

- Customer provides information:
  - risks
  - needs
  - health records
- Provider may sell the information to the insurance company.
- The insurance company can elevate charges!
Message Authentication Codes (MAC)

- Parties $P_1, \ldots, P_N$
- Finite field $\mathbb{F}_p$ with characteristic $p$.
- Unknown and random MAC key $\beta \in \mathbb{F}_p$ is additively shared:
  \[ \beta = \beta_1 + \cdots + \beta_N \]
- MAC (Constraint):
  \[ \gamma(a) = \beta \cdot a \]

### Definition (Shared Secret with MAC)

The secret value $a \in \mathbb{F}_p$ is $\langle \cdot \rangle$-shared between the parties, if each party $P_i$ holds the tuple $(a_i, \gamma(a_i))$, where as

\[
  a = a_1 + \cdots + a_N \\
  \gamma(a) = \gamma(a_1) + \cdots + \gamma(a_N)
\]

### Example (Two parties share $\langle a \rangle$)

- Parties $P_1, P_2$
- MAC key $\beta = -4$
- Finite field $\mathbb{F}_{11} = \{-5, \ldots, 0, \ldots, 5\}$

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>-2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$\gamma(a)$</td>
<td>3</td>
<td>-4</td>
<td>-1</td>
</tr>
</tbody>
</table>
Addition and constant multiplication

Finite field $\mathbb{F}_{11} = \{-5, \ldots, 0, \ldots, 5\}$, MAC key $\beta = -4$

Example (Addition $\langle a \rangle + \langle b \rangle$)

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>-2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$\gamma(a)$</td>
<td>3</td>
<td>-4</td>
<td>-1</td>
</tr>
</tbody>
</table>

$+\quad$

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>5</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>$\gamma(b)$</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a + b$</td>
<td>3</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>$\gamma(a + b)$</td>
<td>-4</td>
<td>-4</td>
<td>3</td>
</tr>
</tbody>
</table>

Example (Multiplication with public constant $e \cdot \langle a \rangle$)

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>-2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$\gamma(a)$</td>
<td>3</td>
<td>-4</td>
<td>-1</td>
</tr>
</tbody>
</table>

$\cdot 2 =\quad$

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e \cdot a$</td>
<td>-4</td>
<td>-1</td>
<td>-5</td>
</tr>
<tr>
<td>$\gamma(e \cdot a)$</td>
<td>-5</td>
<td>3</td>
<td>-2</td>
</tr>
</tbody>
</table>
Multiplication and Input

- Parties use pre-processing data for more complex arithmetic.
- Multiplication $\langle x \rangle \cdot \langle y \rangle$ with triple $\langle c \rangle = \langle a \rangle \cdot \langle b \rangle$:
  1. $\langle \varepsilon \rangle := \langle x \rangle - \langle a \rangle$
  2. $\langle \rho \rangle := \langle y \rangle - \langle b \rangle$
  3. Partially open $\varepsilon$ and $\rho$.
  4. Compute $\langle x \rangle \cdot \langle y \rangle := \langle c \rangle + \varepsilon \cdot \langle b \rangle + \rho \cdot \langle a \rangle + \varepsilon \rho$.
- $P_i$ enters private value $x_i$ with mask value $\langle r \rangle$:
  1. Partially open $\langle r \rangle$ to $P_i$.
  2. $P_i$ sends $\varepsilon := x_i - r$ to all other players.
  3. Compute $\langle x_i \rangle := \langle r \rangle + \varepsilon$.
- Similar for squares.
- Pre-shared random bits are used for comparisons, division, truncations ...
SPDZ Compiler

- Execution Environment
  - Python Library
    - compiler instructions
  - SPDZ Library
    - util instructions
  - Branching
    - if, loops, threads
  - Types
    - int, array, fix, float
  - Registers
    - s, c
  - CISC
    - *, ÷, <
  - RISC
    - +, −, open, input, jump

Source

Compiler

Python

Bytecode & Schedules

*.mpc

Tape Recorder

*.bc

*.sch
Approximate $k$-nearest-neighbour predictions

Parameter Estimation

- 671 users, 9125 items from the MovieLens dataset
- Tested with Baseline
- $k = 9$ and $f' = 4$ are appropriate parameters.