Chapter 3: Approaches for Private Computation

Lecture PETs4DS:
Privacy Enhancing Technologies for Data Science

Dr. Benjamin Heitmann and Prof. Dr. Stefan Decker
Informatik 5
Lehrstuhl Prof. Decker

This is a decorative image and not a logo.

RWTH AACHEN UNIVERSITY

The SPDZ Protocol for SMPC with Active Security

SPDZ vs. CEPS

- What does CEPS do?
  - Provides passive Security
  - But what if parties change the result ?
  - No active Security

- Design Goals of SPDZ
  - Efficient
  - Small Communication Overhead
  - Robust
  - Easy to set up
  - Secure against Malicious parties

SPDZ – An Active Secure Protocol

SPDZ-Overview

- Commonly used SMPC protocol
- Divided in Offline and Online Phase
- Offline Phase
  - Parties share needed information for the function evaluation
  - Multiplication Triples for Multiplication Operations
  - MAC values for the correctness check
  - Parties communicate with each other, so this phase is “still online”, but not part of the actual calculation
- Online Phase
  - Evaluation of the function
- Security guarantee holds even if malicious majority (everybody malicious except for one player)

SPDZ – An Active Secure Protocol

Offline Phase

- Generate Multiplication Triples
  - Also called “Beaver Triples”
  - Consist of three numbers
  - $\langle a \rangle, \langle b \rangle, \langle c \rangle \Rightarrow a * b = c$

- Generate MAC Key
  - “Message Authentication Code”
  - Connected to the performed operations
  - Every party holds a part of the MAC Key
  - After the evaluation some properties still have to be satisfied for correctness

SPDZ – An Active Secure Protocol

Additive Shares

Given n parties $P_1, \dots, P_n$ and a random value a, the value a is called additive shared between the parties when the following holds:

1. Every party holds a shared value a_i
2. $\mathrm{a} = a_1 + \dots + a_n$
3. Party P_i has no information about a_j , when $\mathrm{j} \neq i$

- The shares are assumed to be are random and evenly distributed.
- The protocol for additive sharing of values, is outside the scope of the lecture.

SPDZ – An Active Secure Protocol

MAC Key

Consider n parties $P_1, \dots, P_n$ and a random number α NOT known to any party, called the MAC Key (”Message Authentication Code” Key).

The MAC Key is additive shared along the parties such that the following holds:

$$\alpha = \alpha_1 + \dots + \alpha_n$$

Note that every party P_i only holds α_i, and does not know α.

SPDZ – An Active Secure Protocol

MAC Value

Given n parties $P_1, \dots, P_n$ and a fixed MAC Key α.

Then the MAC value $\gamma(a)$ can be additively shared in a finite field $\mathbb{F}_p$ along the parties such that the following holds:

1. $\gamma(a) = a * \alpha = \gamma(a)_1 + \dots + \gamma(a)_n$
2. Party P_i has no information about $\gamma(a)$ and $\gamma(a)_j$, when $j \neq i$
3. For a *public constant* c party P_i can easily compute a MAC value $\gamma(c)_i = \alpha_i * c$
   - note: party P_i can reconstruct c

Intuition about MAC values: We will use this property of $\gamma(a)$ to verify that no party has modified the results of its local computations. This enables active security.

Notations

Representation of Values and MAC's

- In the **online** phase each shared value $a \in \mathbb{F}_p$ is represented as follows

$$\langle a \rangle = \left((a_1, \dots, a_n), (\gamma(a)_1, \dots, \gamma(a)_n) \right)$$

$$where, a = a_1 + \dots + a_n \; and \; \gamma(a)_1 + \dots + \gamma(a)_n = \alpha * a$$

- Every Party P_i holds $(a_i, \gamma(a)_i)$ and a share of the MAC value α_i.
- The interpretation is that $\gamma(a) = \gamma(a)_1 + \dots + \gamma(a)_n$ is the MAC authenticating a under the global key α.
- Both the shares a_i, and the corresponding MAC value α_i have been pre-computed in the **offline** phase!

SPDZ – An Active Secure Protocol

Offline Phase-Generation of Multiplication Triples

- MAC Key
  - Generation of the MAC key is the first step during execution
  - Parties agree on a MAC key they will use to MAC their data
  - No individual party will actually know the full MAC key
  - The consistency of the result is checked using the MAC key
  - Any manipulated intermediate result can be determined that way

- Generate Multiplication triples using homomorphic encryption (HE)
  - Parties can perform passive secure MPC using HE
  - Malicious behavior possible
  - Idea to mitigate malicious behavior: “Sacrifice” Multiplication triple to check correctness of another triple (explained on the next slides)
    - Doubles the amount of multiplication triples required

SPDZ – An Active Secure Protocol

Online Phase – Addition

- Every party holds
  - additive share of $\langle a \rangle = (a_i, \gamma(a)_i)$
  - additive share of $\langle b \rangle = (b_i, \gamma(b)_i)$
- Every party computes
  - $a_i + b_i = c_i$
  - $\gamma(a)_i + \gamma(b)_i = \gamma(c)_i$
- Note this is a local operation and that we end up with
  - $c = \sum c_i = \sum(a_i + b_i) = (\sum a_i) + (\sum b_i) = a + b$
  - $\alpha * c = \sum \gamma_i(c) = \sum\big(\gamma_i(a) + \gamma_i(b)\big) = \alpha * a + \alpha * b = \alpha * (a + b)$

SPDZ – An Active Secure Protocol

Online Phase – Multiplication by Scalar

- Every party holds
  - additive share of $\langle a \rangle = (a_i, \gamma(a)_i)$
  - Public constant c

  - Every party computes
    - $a_i * c = c_i$
    - $\gamma(a_i) * c = \gamma(a_i * c)$

Online Phase – Addition by Constant

- Every party holds
  - Additive share of $\alpha = \alpha_i$
  - additive share of $\langle a \rangle = (a_i, \gamma(a)_i)$
  - public constant c
- Add public constant c to $\langle a \rangle$
- $\langle a \rangle + c = \big((a_1 + c, a_2 \dots, a_n), (\gamma(a)_1 + c * \alpha_1, \dots, \gamma(a)_n + c * \alpha_n) \big)$
- Note:
  - $a + c = (a_1 + \dots + a_n) + c$
  - $\gamma(a + c) = \gamma(a)_1 + \dots + \gamma(a)_n + c * \alpha = \gamma(a)_1 + \dots + \gamma(a)_n + c * \alpha_1 + \dots + c * \alpha_n$
    $= (\gamma(a)_1 + c * \alpha_1) + \dots + (\gamma(a)_n + c * \alpha_n)$

SPDZ – An Active Secure Protocol

Online Phase – Multiplication

- Every party holds
  - additive share of $\langle a \rangle = (a_i, \gamma(a)_i)$
  - additive share of $\langle b \rangle = (b_i, \gamma(b)_i)$

- To **open** a value $\langle a \rangle$ means: Every party sends their share a_i of $\langle a \rangle$ to all other parties, so that every party can recover the actual value of a.
  This does include sending $\gamma(a)_i$

- To **partial open** a value $\langle a \rangle$ means:
  Every Party sends their share a_i of $\langle a \rangle$ to P_i who computes $a = a_1 + \ldots + a_n$
  This does NOT include sending the respective $\gamma(a)_i$ as well!
  After that P_i broadcasts a to all other Parties

SPDZ – An Active Secure Protocol

Online Phase – Multiplication

- Use multiplication triple generated in the offline phase

$$\langle x \rangle, \langle y \rangle, \langle z \rangle \Rightarrow x * y = z$$

- For each multiplication a new triple is needed

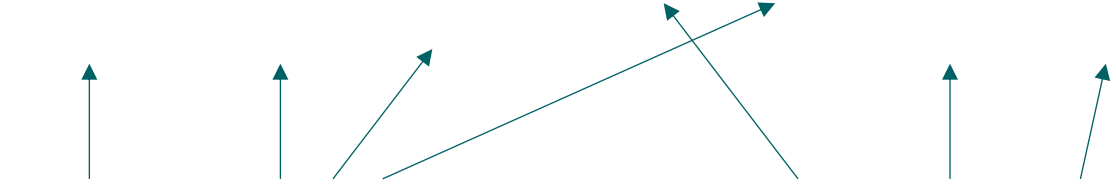
- We want to compute $\langle a * b \rangle$ of the shared values $\langle a \rangle, \langle b \rangle$
  - Every party **"partially opens" masked values of *a* and *b*.**
    - $\epsilon = a - x$
    - $\delta = b - y$

  - After this, $\langle a * b \rangle$ can then be computed via the following local operation:

$$\langle a * b \rangle = \langle z \rangle + \langle y \rangle * \epsilon + \langle x \rangle * \delta + \epsilon * \delta$$

Online Phase – Multiplication

$$\langle a * b \rangle = \langle (a - x + x) * (b - y + y) \rangle$$
$$= (a - x) * (b - y) + (b - y) * \langle x \rangle + (a - x) * \langle y \rangle + \langle x * y \rangle$$



Opened by the parties

Multiplication Triple

$$= \epsilon * \delta + \delta * \langle x \rangle + \epsilon * \langle y \rangle + \langle z \rangle$$

Please keep in mind that ϵ and δ refer to the masked values of a and b in this formula.

How to share the Input of Party P_i

- Problem: Party P_i wants to additively share its input a
- This means we want to compute $\langle a \rangle = \left((a_1, \dots, a_n), (\gamma(a)_1, \dots, \gamma(a)_n) \right)$ such that:
  - $a = a_1 + \dots + a_n$
  - $\gamma(a) = \gamma(a)_1 + \dots + \gamma(a)_n$
- Assume offline phase provides a random $\langle r \rangle$.
  - $\langle r \rangle$ is opened to P_i: This means only P_i knows r
- P_i broadcasts $\Delta = a - r$
  - now a is masked with a value only P_i knows
- Then all players compute $\Delta + \langle r \rangle = \langle a \rangle$
  - Now all players hold a share $(a_j, \gamma(a)_j)$ without knowing the actual value of a
  - This is an Addition with a public constant (see the relevant slide for details)
- **Note:** This method shares a additively in a secure way and computes MAC values for a based on the MAC values of r ! In this way, pre-computed r_i and $\gamma(r)_j$ from the offline phase are used to share a in the on-line phase!

Verifying Correctness at the end of the circuit evaluation – Part 1

- Each party i has an agreed set of t partially opened values

$$\Omega_j, \qquad 1 \leq j \leq t$$

- and each one has a sharing of the associated MAC value

$$\gamma(\Omega_j)_i, \quad 1 \leq j \leq t$$

- Each party i also has a share of the MAC key

$$\alpha_i.$$

Verifying Correctness at the end of the circuit evaluation – Part 1

- Each party i computes

$$\Omega = \sum_{j=1}^{t} \Omega_i$$

- They also compute their share of the MAC on Ω

$$\Gamma_i = \sum_{j=1}^{t} \gamma(\Omega_j)_i$$

- And then

$$\sigma_i = \Gamma_i - \alpha_i * \Omega$$

- Note, if all is correct then σ_i is a sharing of zero
  - Then all check whether $\sigma_1 + \ ... + \sigma_n = 0$ (by broadcasting σ_i)

Summary of online phase for SPDZ protocol

1. **Initialisation:** The parties agree on MAC key α, a set of multiplication triples and a set of random values to use. All of these values have been pre-computed and pre-shared in the offline phase, and no party knows the actual values corresponding to these shared values.
2. **Input:** Each party shares its inputs to all other parties as described earlier.
3. **Computation:** The circuit is evaluated using addition and multiplication.
4. **Output:** The output of the circuit is recovered and all parties verify the correctness of the result and all partially opened values.

- **The details of the offline phase of SPDZ are not covered in the lecture**

Complexity of SPDZ

- Offline Phase
  - Expensive part of the calculation
  - Generation of the Multiplication Triples has to be done with HE
  - Correctness check of triples consumes triple
  - Check if $\langle x \rangle * \langle y \rangle = \langle z \rangle$ holds using triple and MAC
  - $O(n^3)$

- Online Phase
  - All Operations can be calculated in $O(n)$
  - Produces Overhead of $O(n)$ + complexity of the arbitrary function

Improvements of SPDZ: MASCOT and SPDZ 2.0

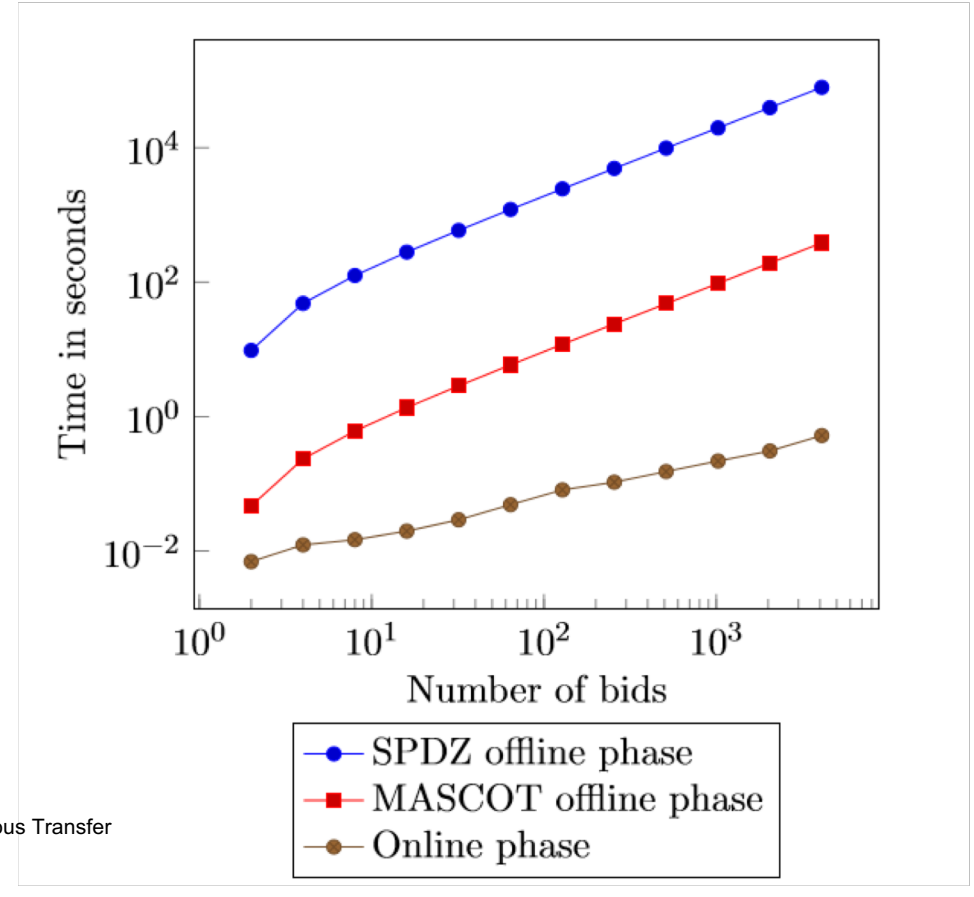
MASCOT

- The offline phase is a bottleneck of the SPDZ protocol
- FHE for the multiplication triple generation is expensive
  - $O(n^3)$
- MASCOT tries to improve offline phase
  - Oblivious Transfer for Multiplication Triple Generation is much faster
  - FHE isn't needed anymore
  - Parallelism of generation possible
- **Online phase stays the same**

SPDZ vs MASCOT

- Implementation of Vickrey auction
  - Every party sends bid privately
  - Party with highest bid wins, but only has to pay second highest bid

- MASCOT offline phase is ~ 200 times faster



MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer
https://eprint.iacr.org/2016/505.pdf

SPDZ 2.0

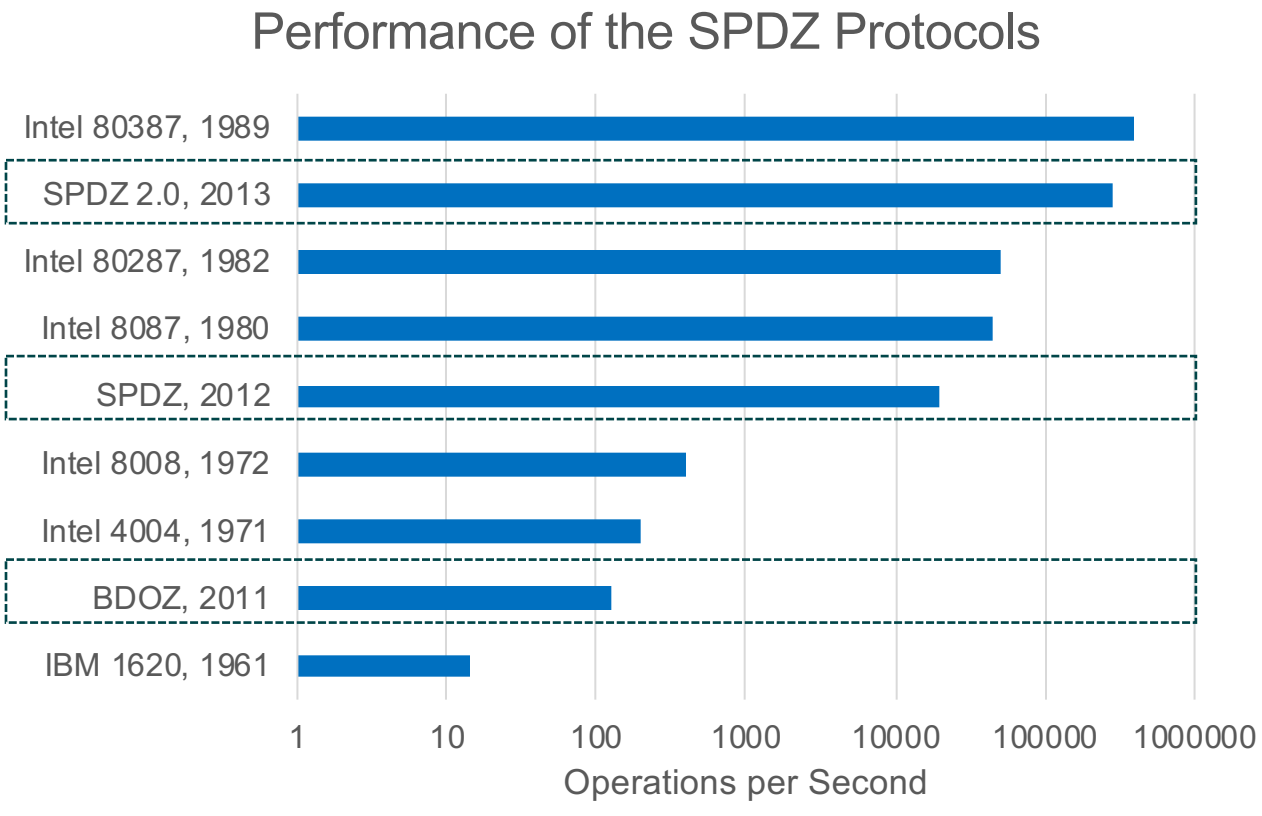
- Reduction of shares is done by a *program dependency graph* $G = (V, E)$
  - Vertices relate to instructions
  - Edges represent flow between instructions
  - Merge nodes which don't have any preconditions
  - Merged instructions can be done in one open sharing procedure

- Off-line phase not just for multiplication triples but also for pre-sharing of secrets for other operations:
  - Square $b = a^2$
  - Inverse $a^{-1} = a$

- Rearrange Operations
  - $a \oplus b = (a - b)^2$ is expensive normally
  - Through bit decomposition only one open sharing is needed, because $a \oplus b = a + b - 2 * a * b$ in binary

- Polynomial approximation of $\sin(x)$ can be reduced from 5874 OS to 759 OS

SPDZ 2.0



Based on : https://www.cs.bris.ac.uk/home/ps7830/spdz2.pdf

Summary of SPDZ and Active Security for SMPC

Discussion

- Advantages
  - Secure Multiparty Computations works fine
  - Open source implementations available e.g. on github
  - Detection of malicious behavior even if n-1 parties are malicious
  - Fixed settings might be practicable since the offline phase can be run in background

- Disadvantages
  - Bad Scaling
  - Computational power of the 80s
  - How to reuse results ? If one party is malicious the whole calculation gets worthless
  - Changing settings require new preprocessing

⇒ Usage of SMPC in practical applications is currently limited due to the overheads in performance and scalability, but future potential is huge!

Secure Multi-party Computation: past thesis topics 1/2

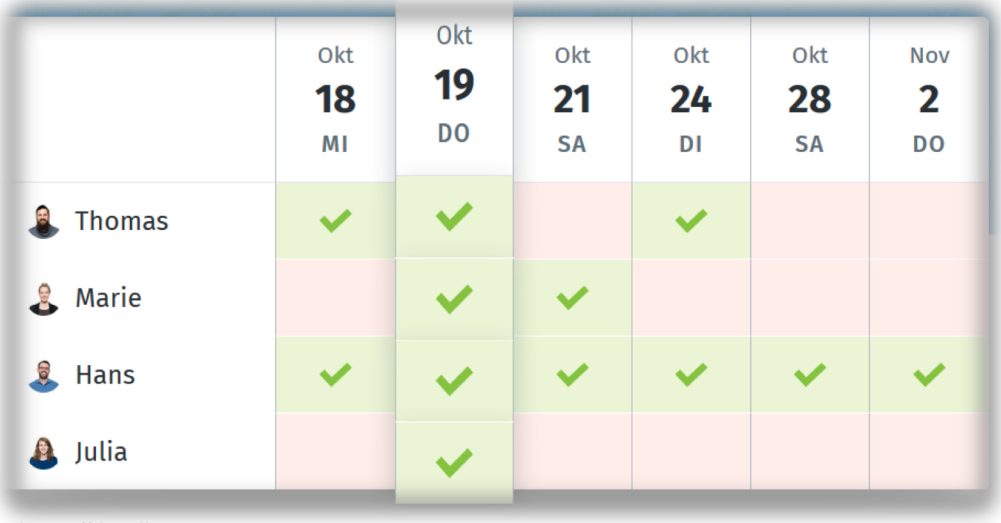
Master thesis:
Jonas Nagy-Kuhlen
Interactive Private Multi-Party Calendar Scheduling

RWTH AACHEN UNIVERSITY

Motivation

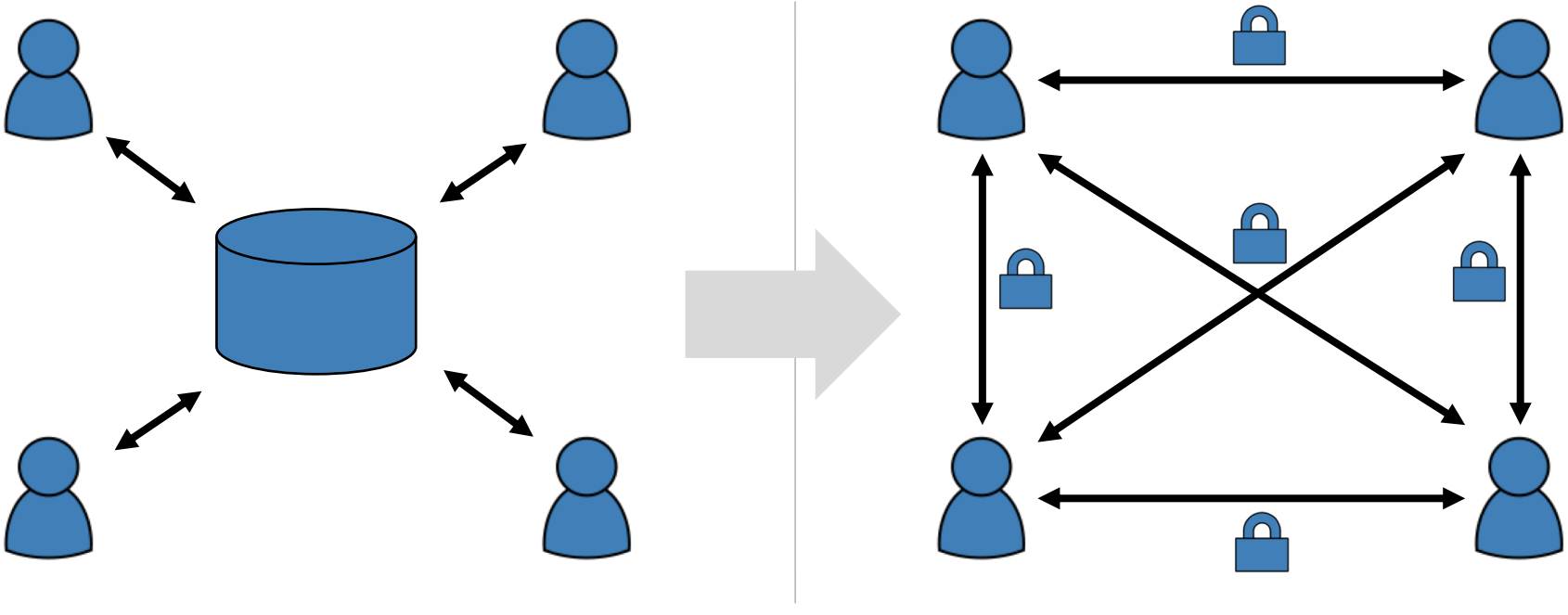
- Popularity of calendar scheduling applications
- Appointments can be arranged easily

→ Increased productivity for companies and individuals

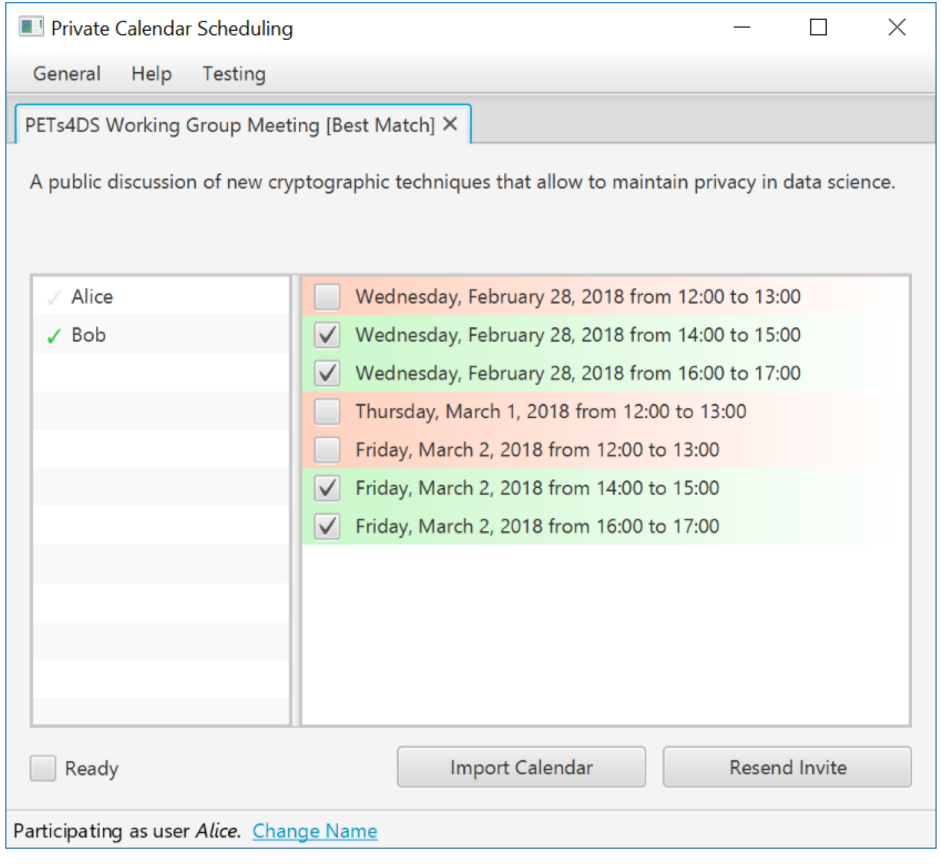


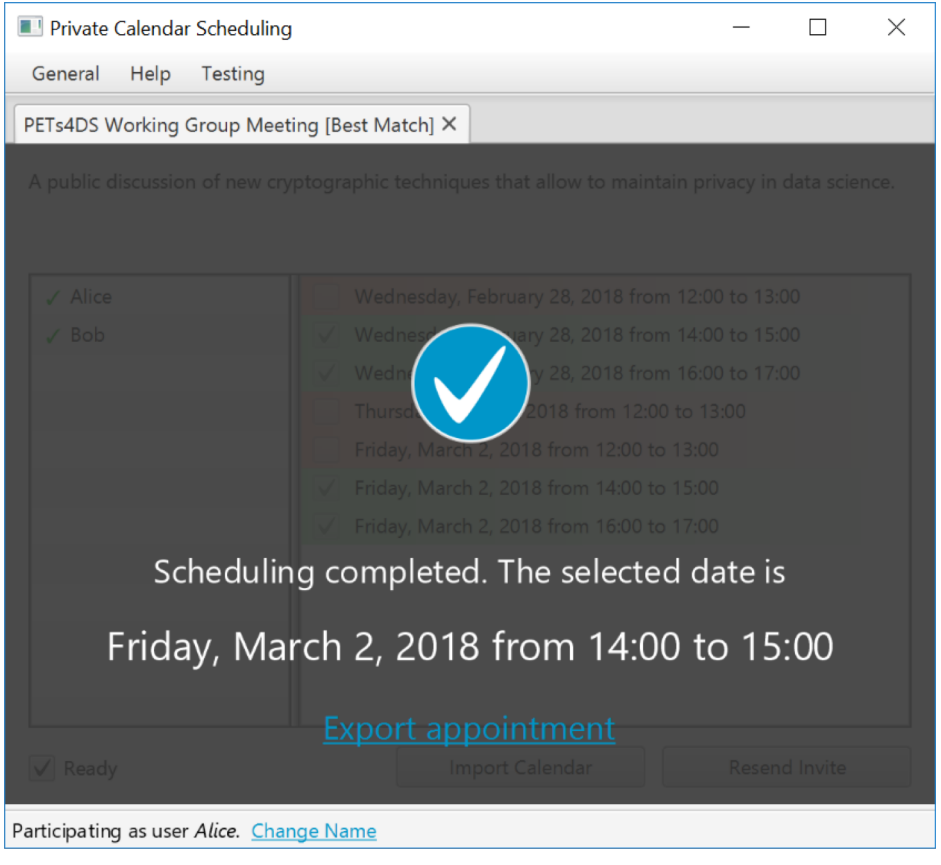
https://doodle.com

Can we do better?

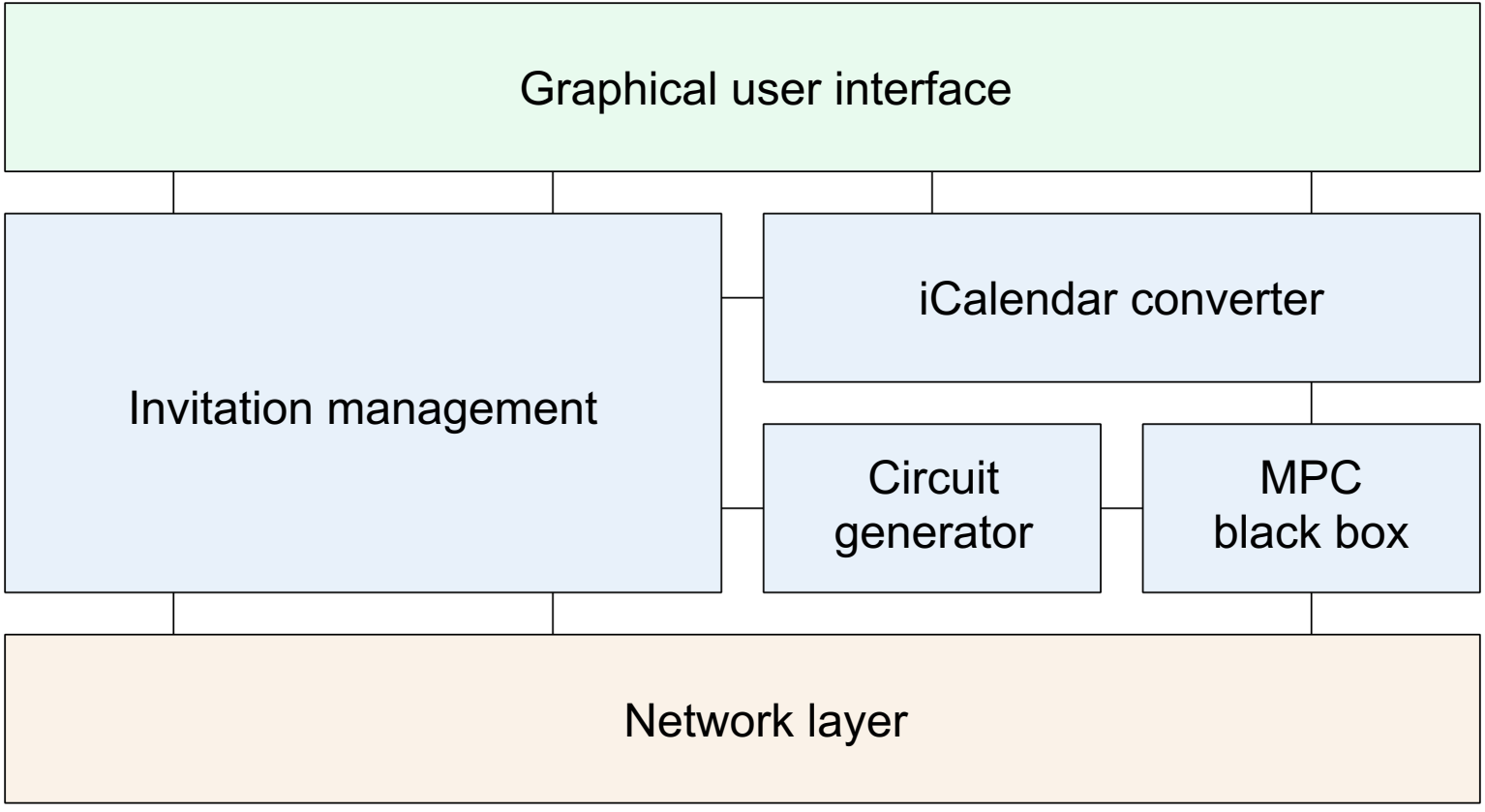


Goal: Development of a Secure Calendar Application





High-Level Components

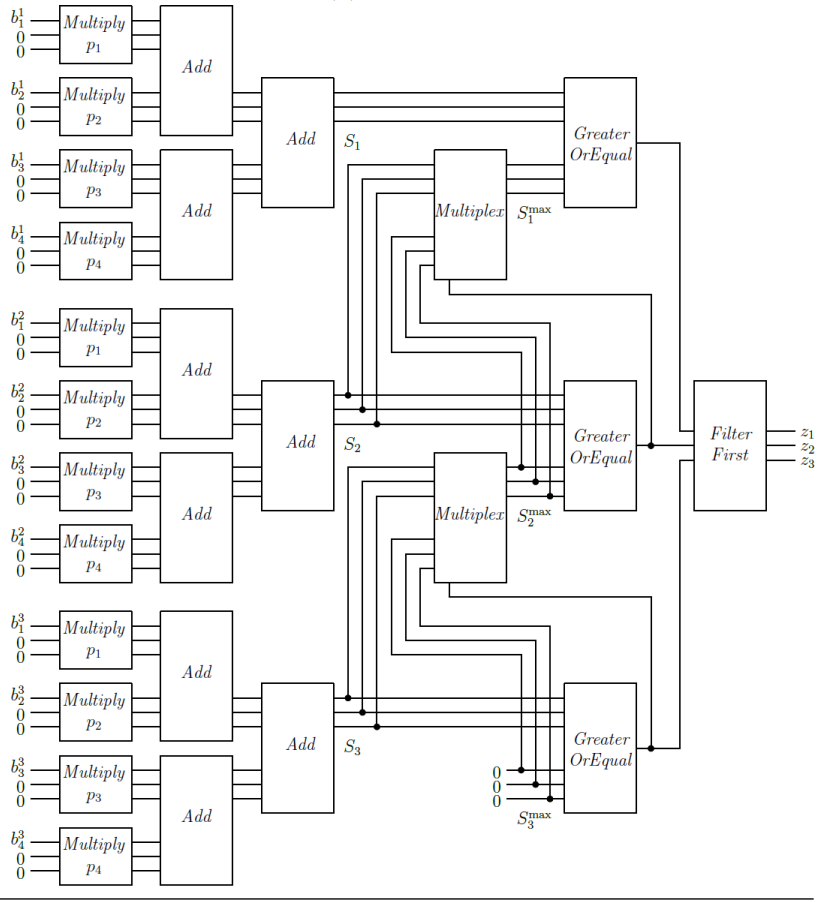


Circuit Construction

Weighted best match

- Integer addition allows to accumulate scores for each time slot
- Comparison and multiplexing blocks allow to find index with highest score

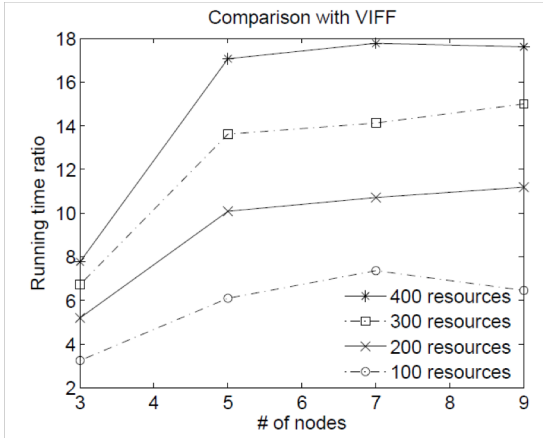
Number of parties $n = 4$
Number of time slots $m = 3$

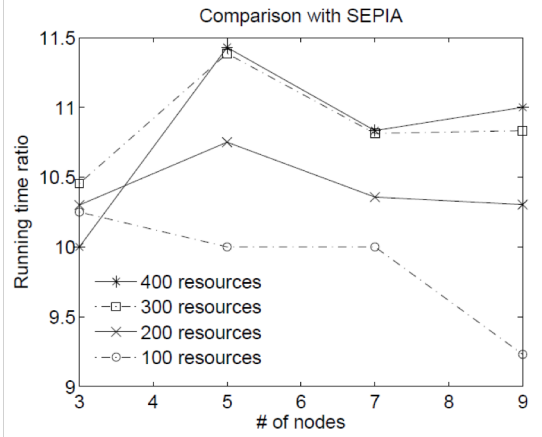


Why use SCAPI as MPC library?

- GMW protocol is secure in semi-honest setting
  - Security against malicious adversaries comes with additional costs
- Boolean circuits considered more flexible than arithmetic circuits in this use case
  - Only small fixed-size integer logic required
- Well-documented and well-structured
  - Can be easily integrated into Java code

"Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces", Choi et al., 2012



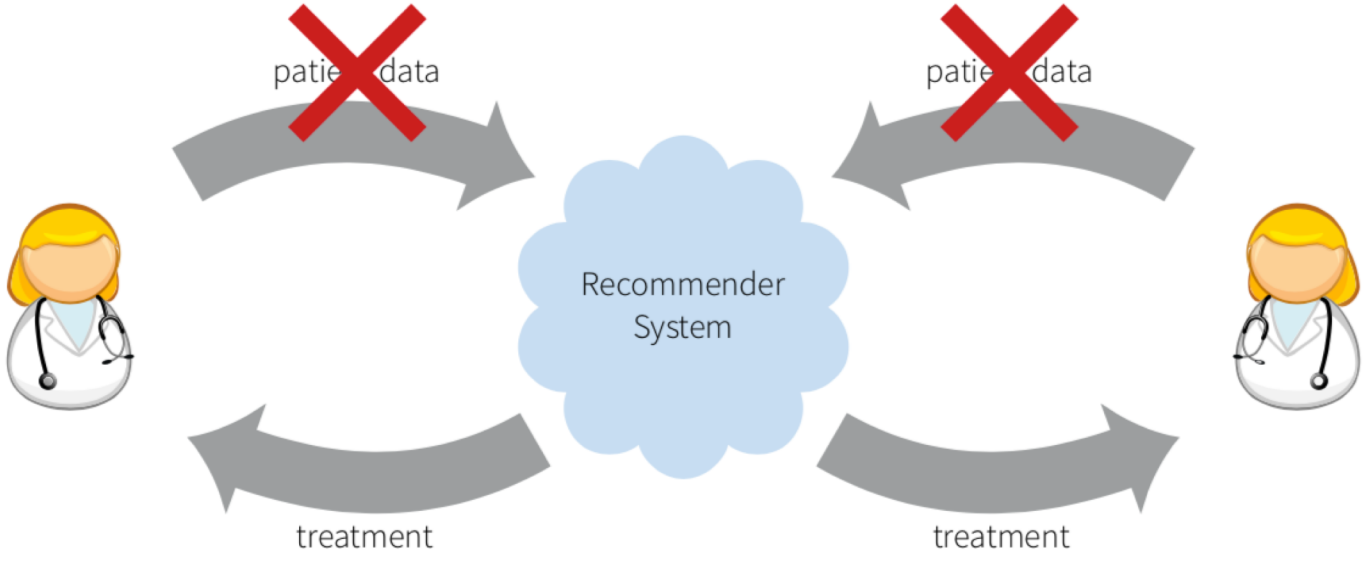


Secure Multi-party Computation: past thesis topics 2/2

Master thesis:
Thibaud Kehler
Privacy-preserving collaborative filtering with SPDZ

Use Case

Medical Data



Patient data is confidential by law!

Threat Analysis

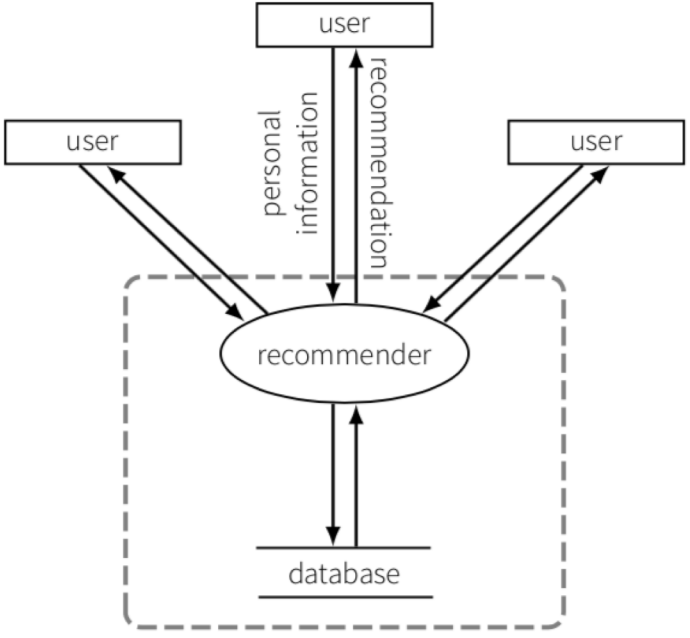
Items of Interest (IOI)

- 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?

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.

Build Model
- Plain Ratings $\Theta(n^2 \cdot m)$
- Sparse Ratings $\Theta(n^2 \cdot c^2)$

Approx. k-NN Predictions $\Theta(f' \cdot n)$

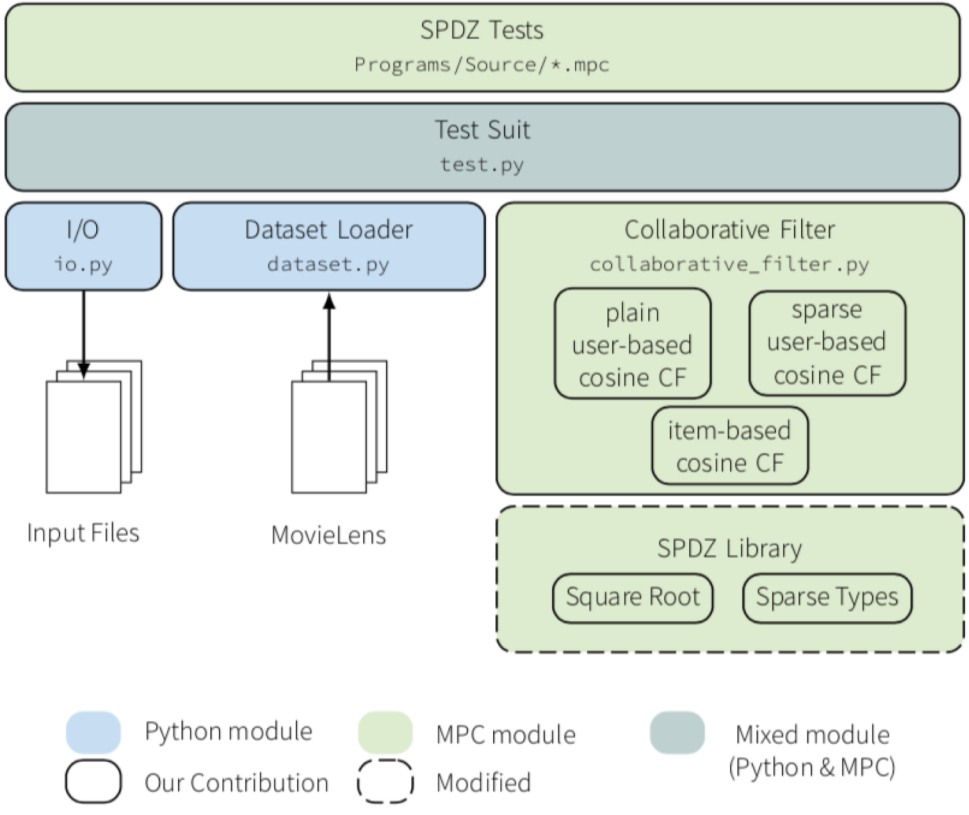
Item-Based Model

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 $\Theta(m^2 \cdot n)$

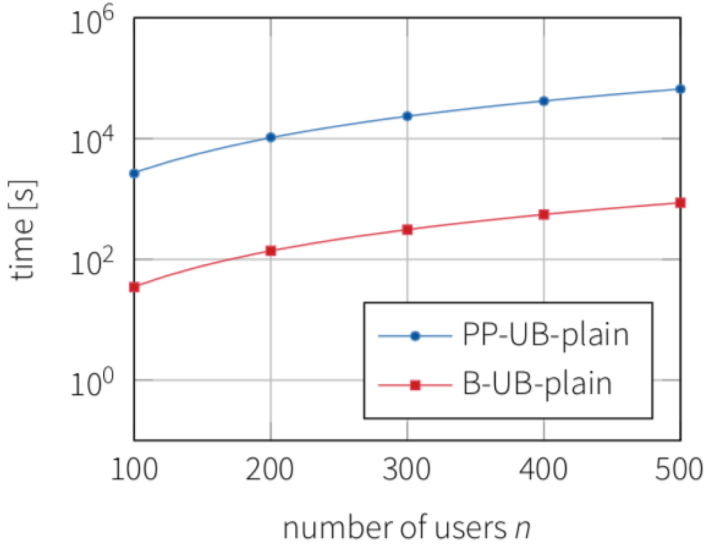
k-NN Predictions Computed locally

Implementation



Performance

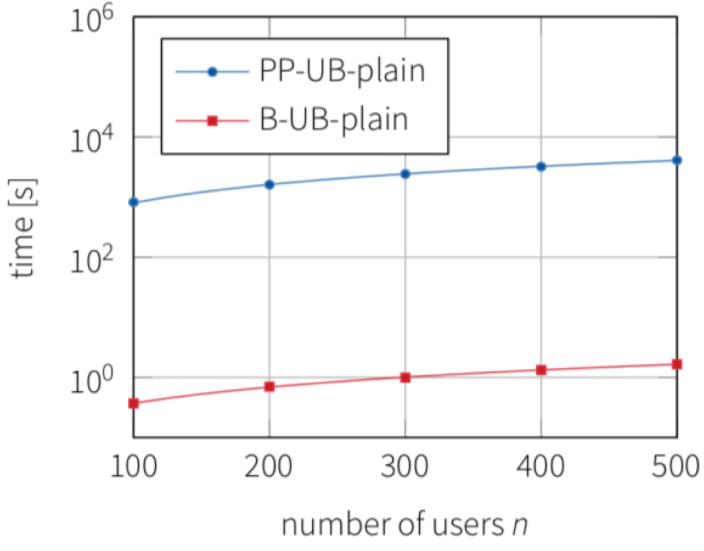
User-Based Model



- 3000 items
- ≈ 78 **times slower**

Performance

User-Based Predictions



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

Summary of Chapter 3

Comparison of Approaches for Private Computation

- **Homomorphic Encryption (HE)**
  - Encrypt data before sending to cloud
- **Secure Multi-Party Computation (SMPC)**
  - Compute results together without sharing private data

Comparison of Approaches for Private Computation

Approach	Adversary Type	Confidentiality	Integrity	Requires Interaction
Homomorphic Encryption (HE)	Honest-but-curious (HBC)	YES	NO	NO
Secure Multi-Party Computation (SMPC)	Honest-but-curious (HBC) or Malicious	YES	YES	YES

Homomorphic Encryption: Overview of Properties

- **Confidentiality:** yes
  - Compute and storage nodes do not have access to the unencrypted data
- **Integrity:** no
  - Modification of data is possible
  - Faulty computation is possible
- **Adversary type addressed:** honest-but-curious adversaries
  - No party can learn anything from the other party
  - But no guarantee about malicious behavior of any party
- **Requires interaction:** no
  - HE is possible with only one party, with two parties and with three or more parties
  - However, the same key has to be used by all parties wishing to share results

Secure Multi-Party Computation : Overview of Properties

- **Confidentiality:** yes
  - No node learns anything new, besides the result of the computation.
- **Integrity:** yes
  - Intermediate results have to be shared, which allows verification of correctness.
- **Adversary type addressed:** malicious and HBC
  - Most SMPC schemes allow $t < \frac{n}{2}$ HBC adversaries or $t < \frac{n}{3}$ malicious adversaries, but higher bounds also possible.
- **Requires interaction:** yes
  - SMPC usually requires 3 nodes or more.
  - Confidentiality and integrity of SMPC are enforced through interaction between nodes.

Business use case for computation on secret data

Show me the money!

- Important applications of Data Science:
  - Trends / forecasting
  - Clustering
  - Anomaly detection

- Common aspects:
  - Aggregation of data from multiple parties
  - Computation adds value to existing data
  - Computation requires sharing of data
  - Sensitive nature of input **and** output data

- Current approach: centralised computation
  - Requires trusting central party
  - Easy if everybody on the same team
  - Problem for multiple companies / jurisdictions / countries

- Whenever the participants do not trust each other, but the data analytics result is worth a lot of money, and processing time is unimportant, then SMPC and HE should be considered.

Revisiting the goals of this chapter

Two approaches for computation with secret data:
- Homomorphic encryption (HE)
- Secure multi-party computation (SMPC)

Multiple implementations for each approach:
- HE: RSA, Paillier Cryptosystem
- SMPC: YGC, CEPS, SPDZ

Differences:
- Number of parties
- Complexity overhead
- Different guarantees (confidentiality, integrity, interaction)

Challenges: Computational overhead + architecture!