Indian Institute of Technology, Kanpur
Computer Science and Engineering
Assignment 1
CS670: Cryptographic Techniques for Privacy Preservation

Instructor: Adithya Vadapalli

28/08/2025

This assignment is long! Please start early. Doing this assignment correctly is very important, as every other assignment will be built on this.

Deadline: September 26th, 2025, EOD on Hello IITK. No Extension will be given.

Academic Integrity You have to do the assignment individually. You are encouraged to ask the instructor for help. Sign up on Piazza to ask for help: `https://piazza.com/iitk.ac.in/firstsemester2025/cs670`. Students can should come to the Instructor's office hours for help in coding. However, copying code from others or using AI tools without understanding the solution is not allowed and will result in 0 marks for the assignment. Students may be asked to explain their code and solution during evaluation.

Background In recommendation systems, users and items are represented by latent vectors:

$$U \in \mathbb{R}^{m\times k}, \quad V \in \mathbb{R}^{n\times k}.$$

For the purposes of the assignment, assume that:

$$U \in \mathbb{Z}^{m\times k}, \quad V \in \mathbb{Z}^{n\times k}.$$

1. m is the number of users.
2. n is the number of items.
3. k is the number of features.

A user i's profile u_i and an item j's profile v_j interact to produce a prediction

$$\hat{r}_{ij} = \langle u_i, v_j \rangle.$$

After observing a query, user profiles are updated according to:

$$u_i \leftarrow u_i + v_j \left(1 - \langle u_i, v_j \rangle\right).$$

In this assignment, U and V are secret-shared among parties. The task is to implement secure updates without revealing the underlying vectors.

Objective The main objective of this assignment is to implement secure user-profile updates. $\mathcal{S}_0$ and $\mathcal{S}_1$ hold additive shares of U and V:

$$U = U_0 + U_1, \quad V = V_0 + V_1$$

Suppose user i queries item j. Then we have to execute the following steps:

1. Compute the inner product $\langle u_i, v_j \rangle$ in MPC.
2. Compute $\delta = 1 - \langle u_i, v_j \rangle$ securely.
3. Update $u_i \leftarrow u_i + v_j \cdot \delta$.
4. Re-share the updated u_i among $\mathcal{S}_0$ and $\mathcal{S}_1$.

Implementation Hints

1. Generate secret-shared vectors for the parties (e.g., via a program `./gen_queries` that produces share files). A real client is not required.
2. Each party reads the relevant line from its file to obtain the secret shares corresponding to the first query.
3. Derive shares of v_j securely. Consult the instructor if unsure.
4. Once shares of both u_i and v_j are obtained, perform a secure dot product in MPC.

Implementation Suggestions

1. Create a function `MPC_DOTPRODUCT`.
2. Define a structure for shares in `shares.h` for easy initialization and randomization.
3. Include a `randomize()` function to initialize shares randomly.
4. For $\mathcal{S}_0$ and $\mathcal{S}_1$, the same program can be used with different role definitions.

Deliverables There is some helper code on communication in C++ and Docker here: `https://github.com/privacy-iitk/cs670-25-Monsoon/tree/main/A1`.

- Your submission must include a Dockerfile and docker-compose.yaml file.
- The command line arguments that your program should take are the number of users, the number of items and features, and the number of queries.
- The code should print the secret shares of P0 and P1, and the reconstructed value (for debugging only).
- README clearly explaining how to run the code.
- A short report as part of README (1–2 pages) explaining:
  - Secret-sharing implementation.
  - How inner products and updates were computed securely.
  - Communication rounds and efficiency considerations.

Grading

- Correctness and Security of updates in additive MPC: 80%
- Code clarity and documentation: 20%
- Bonus: replicated MPC implementation: 10%