CS670: Cryptographic Techniques for Privacy Preservation
Assignment 3

Shubham Rawat (251110070)
Department of Computer Science and Engineering, IITK

Secret Sharing Implementation

The Matrix U and V are split into additive shares between two parties S_0 and S_1:

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

Each party stores its own share locally and never reveals it to others.

To update user i, the protocol needs two values:

- The user index i — public.
- The item index j — private.

The item index is represented as a secret-shared standard basis vector of size equal to the number of items. This enables privately selecting v_j during MPC.

Once both shares of u_i and v_j are handled, the following secure update is performed:

$$\delta = 1 - \langle u_i, v_j \rangle$$

$$u_i \leftarrow u_i + v_j \cdot \delta$$

All multiplications (vector–scalar, vector–vector, matrix–vector) use Beaver Triplets.

MPC Inner Product and Updates

Secure Dot Product

To compute $\langle u_i, v_j \rangle$ securely:

1. A trusted dealer sends Beaver triplet shares.
2. Parties compute blinded values and exchange them.
3. Each computes its share of the multiplication.

The same method is used for:

- Vector–scalar MPC multiplication
- Vector–vector dot products
- Vector–matrix multiplication

Each party updates its own share of v_i locally.

Distributed Point Function (DPF)

Here for DPF generation we kept the logic same as in Assignment 2. The changes are as follows:

1. For the leaf nodes, instead of using scalar values, we convert them into vectors of size equal to the row length of the item matrix.
2. The final correction word (FCW) is also changed to a vector.
3. The correction words are computed as:

$$\mathrm{FCW}_1 = \mathrm{prgVector}(\mathrm{dist}(\mathrm{genValue}), k)$$

   (a random vector),

$$\mathrm{FCW}_0 = \mathrm{leafVecAtTarget0} - \mathrm{leafVecAtTarget1} - \mathrm{FCW}_1.$$

4. Both parties exchange masked values:

$$M_0 - \mathrm{FCW}_0 \quad \text{(from Party 0)}, \qquad M_1 - \mathrm{FCW}_1 \quad \text{(from Party 1)}.$$

   Then each party computes:

$$\mathrm{fcwm} = (M_0 - \mathrm{FCW}_0) + (M_1 - \mathrm{FCW}_1) = M - \mathrm{FCW}.$$

5. One party multiplies the leaf value by -1 and applies the final correction word based on flag bits (as per DPF definition).
6. At the target index, for Party 0:

$$v_0 = \mathrm{leafVecAtTarget0}, \qquad v_1 = \mathrm{leafVecAtTarget1}.$$

   Applying the correction:

$$v_0 + \mathrm{fcwm} = v_0 + (M - \mathrm{FCW}) = v_0 + M_0 - \mathrm{FCW}_0 - \mathrm{FCW}_1$$

   Since:

$$\mathrm{FCW}_0 = v_0 - v_1 - \mathrm{FCW}_1,$$

   substituting gives:

$$v_0 + M_0 - (v_0 - v_1 - \mathrm{FCW}_1) - \mathrm{FCW}_1 = M_0 + v_1.$$

7. For Party 1 (as per DPF definition), it simply outputs:

$$-v_1$$

   (no FCW is applied on this party).

8. Final reconstruction:

$$(M_0 + v_1) + (-v_1) = M.$$

Communication and Efficiency

- Dealer sends Beaver triplets, shares of 1, and standard vector shares.
- Parties exchange blinded values three times:
  - For computing v_j
  - For $\langle u_i, v_j \rangle$
  - For vector–scalar multiplication
- Boost.Asio coroutines allow fully asynchronous message passing.

Code Structure

- `shares.h`: Structure definitions for secret shares.
- `mpcOperations.h/cpp`: Implements all MPC multiplications.
- `gen_queries.cpp`: Generates Beaver triplets and shared vectors.
- `dealer.cpp`: Coordinates MPC, sends triplets, reconstructs results.
- `party0.cpp`, `party1.cpp`: Perform MPC steps and communicate.
- `common.hpp`: Contains coroutine-based network utilities.

Build and Run Commands

```
./runProtocol.sh
```

Conclusion and Result

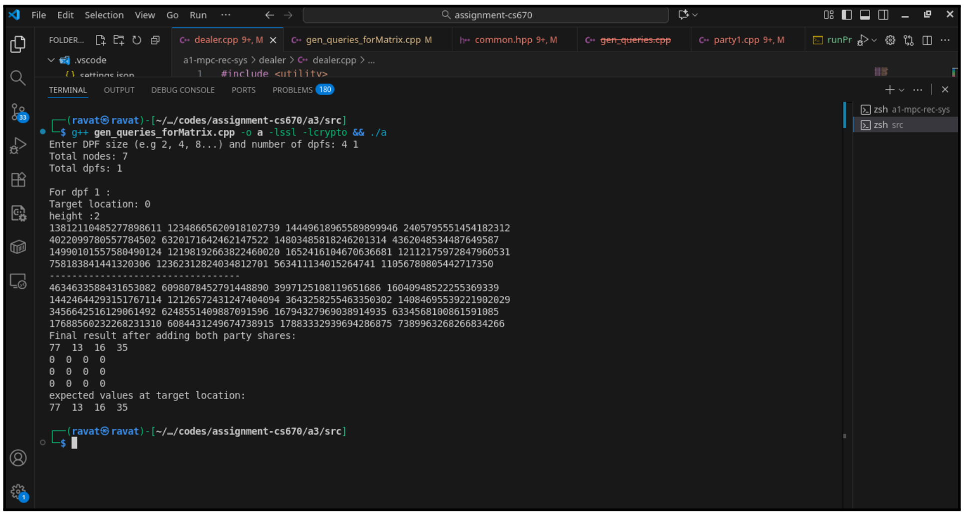

Figure 1: DPF result

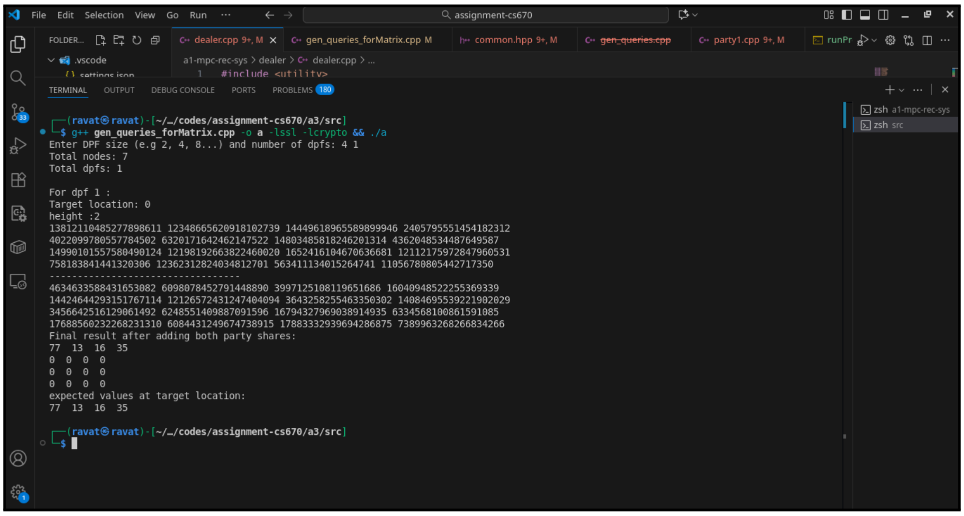

Figure 2: Final Output

This assignment demonstrates secure multiparty computation using additive secret sharing and dpf to perform recommendation updates privately and efficiently.