Task 1: Secure Multiparty Computation using Beaver Triples

Sevak Shekokar

1. Introduction

This document explains the concept, implementation, and detailed working of **Secure Multiparty Computation (MPC)** using **Beaver Triples**. The project simulates a 3-party computation that multiplies secret values without revealing the inputs to any of the participating parties.

2. What is MPC?

Secure Multiparty Computation (MPC) allows a group of parties to compute a function over their inputs while keeping those inputs private. No individual party learns anything about others' private inputs, yet the final output is correct.

Example use cases:

- Privacy-preserving statistics (e.g., average salary)
- Secure voting and auctions
- Federated learning and ML on private data

3. Key Technique: Beaver Triples

Beaver triples are preprocessed random values that enable secure multiplication:

- Random values: a, b
- Their product: $c = a \times b$
- Secret shares of a, b, c are distributed to each party

Secure Multiplication Formula:

$$x \cdot y = d \cdot e + d \cdot b + e \cdot a + c$$

Where:

- $d = x - a$
- $e = y - b$

This trick allows parties to securely compute $x \cdot y$ without revealing x or y.

4. Project Setup and Flow

4.1. Components

- **coordinator.py** – acts as server; distributes shares, orchestrates protocol, reconstructs result
- **party.py** – each party receives shares, performs secure local computation, and sends result back

4.2. Workflow

1. The coordinator secret-shares inputs x and y among 3 parties
2. A Beaver triple $(a, b, c = ab)$ is generated and also shared
3. Each party computes:
$$d_i = x_i - a_i, \quad e_i = y_i - b_i$$
4. The coordinator collects all d_i, e_i and computes:
$$d = x - a = \sum d_i, \quad e = y - b = \sum e_i$$
5. The coordinator computes $d \cdot e$, shares it again among the parties
6. Each party computes:
$$\text{share}_i = \text{de\_share}_i + d \cdot b_i + e \cdot a_i + c_i$$
7. The coordinator collects and sums the 3 shares to get the final result:
$$x \cdot y = \sum \text{share}_i$$

5. Why Coordinator Computes $d \cdot e$

Even though d and e are public, the value $d \cdot e$ is re-shared to prevent all parties from seeing it directly. If they saw the full $d \cdot e$, it would leak intermediate data — violating MPC's goal.

7. Worked Example for Revision

Goal: Securely compute $x \times y = 8 \times 5 = 40$ using 3-party MPC with Beaver triples.

Step 1: Secret Share Inputs

We split $x = 8$ and $y = 5$ into 3 shares each:

$$x_1 = 2, \quad x_2 = 3, \quad x_3 = 3 \Rightarrow x = x_1 + x_2 + x_3 = 8$$

$$y_1 = 1, \quad y_2 = 2, \quad y_3 = 2 \Rightarrow y = y_1 + y_2 + y_3 = 5$$

Step 2: Generate Beaver Triple

Let the Beaver triple be:

$$a = 4, \quad b = 2, \quad c = a \cdot b = 8$$

Shares of a, b, c:

$$a_1 = 1, \quad a_2 = 2, \quad a_3 = 1 \Rightarrow a = 4$$

$$b_1 = 0, \quad b_2 = 1, \quad b_3 = 1 \Rightarrow b = 2$$

$$c_1 = 3, \quad c_2 = 2, \quad c_3 = 3 \Rightarrow c = 8$$

Step 3: Each Party Computes $d_i = x_i - a_i$, $e_i = y_i - b_i$

$$\begin{aligned} d_1 &= 2 - 1 = 1, & e_1 &= 1 - 0 = 1 \\ d_2 &= 3 - 2 = 1, & e_2 &= 2 - 1 = 1 \\ d_3 &= 3 - 1 = 2, & e_3 &= 2 - 1 = 1 \end{aligned}$$

Step 4: Coordinator Reconstructs

$$d = d_1 + d_2 + d_3 = 4, \quad e = e_1 + e_2 + e_3 = 3$$
$$d \cdot e = 4 \cdot 3 = 12$$

Step 5: Re-share $d \cdot e = 12$

Let the shares be:

$$de_1 = 5, \quad de_2 = 4, \quad de_3 = 3$$

Step 6: Each Party Computes Final Share

$$\text{Party 1: } 5 + 4 \cdot 0 + 3 \cdot 1 + 3 = 5 + 0 + 3 + 3 = 11$$

$$\text{Party 2: } 4 + 4 \cdot 1 + 3 \cdot 2 + 2 = 4 + 4 + 6 + 2 = 16$$

$$\text{Party 3: } 3 + 4 \cdot 1 + 3 \cdot 1 + 3 = 3 + 4 + 3 + 3 = 13$$

Step 7: Coordinator Reconstructs Final Result

$$\text{Result} = 11 + 16 + 13 = 40 \quad \text{(Correct!)}$$

This confirms the secure computation of $x \cdot y$ using 3-party MPC and Beaver triples.

6. Final Notes

This system securely computes multiplication without revealing any inputs to individual parties. It can be extended to support:

- Dot product over vectors
- XOR-based secret sharing
- Parallel secure multiplications

Author: Sevak Shekokar
Roll No: 241110065
M.Tech CSE, IIT Kanpur