

# A Survey On Secure Machine Learning

Taobo Liao | Taoran Li | Prathamesh Nadkarni  
taobol2|taoranl2|pyn2

@illinois.edu

University of Illinois Urbana-Champaign  
Urbana, IL 61801

April 22, 2024

## Abstract

In this survey, we will explore the interaction between secure multiparty computation and the area of machine learning. Recent advances in secure multiparty computation (MPC) have significantly improved its applicability in the realm of machine learning (ML), offering robust solutions for privacy-preserving collaborative learning. This review explores key contributions that leverage MPC to enable multiple parties to engage in ML tasks without compromising the privacy of their data. The integration of MPC with ML frameworks facilitates the training and evaluation of models on combined datasets from various sources, ensuring that sensitive information remains encrypted throughout the process. Innovations such as specialized software frameworks and domain-specific languages streamline the adoption of MPC in ML, optimizing performance and broadening its usage. These frameworks address both semi-honest and malicious threat models, incorporating features such as automated optimizations and cryptographic auditing to ensure compliance and data integrity. The collective insights from these studies highlight MPC's potential in fostering collaborative yet confidential data analysis, marking a significant stride towards the realization of secure and efficient computational solutions in privacy-sensitive industries. This paper investigates a spectrum of SecureML libraries that includes cryptographic protocols, federated learning frameworks, and privacy-preserving algorithms. By surveying the existing literature, this paper aims to examine the efficacy of these libraries in preserving data privacy, ensuring model confidentiality, and fortifying ML systems against adversarial attacks. Additionally, the study explores an innovative application domain for SecureML techniques: the integration of these methodologies in gaming environments utilizing ML.

# 1 Introduction

Secure multiparty computation (MPC) has emerged as a pivotal technology enabling privacy-preserving machine learning (ML) across various domains. This review explores three significant contributions to the field:

1. Secure multi-party computation (MPC) is a transformative cryptographic protocol that enables multiple parties to collaboratively engage in machine learning tasks without compromising the privacy of their individual data inputs. This paper [MZ17] delves into the application of MPC in machine learning, showcasing recent advancements and frameworks that facilitate privacy-preserving computations in a variety of applications. We will introduce this part in section 2.
2. The CRYPTEN [KVH<sup>+</sup>21] framework from Facebook AI Research exemplifies a significant stride in making MPC protocols accessible to machine learning practitioners who lack deep cryptographic expertise. CRYPTEN integrates seamlessly with popular machine learning frameworks, using secure MPC primitives to support private model evaluation and training. The framework is designed to work under a semi-honest threat model and utilizes GPU acceleration to enhance performance, demonstrating its utility with state-of-the-art models for text classification, speech recognition, and image classification. It aims to lower the barrier to adoption of MPC in machine learning by providing a user-friendly interface that mimics conventional machine learning APIs while ensuring secure computations. We will introduce this part in section 3.
3. Another pivotal development is Cerebro [ZDC<sup>+</sup>21], a comprehensive platform for MPC-based collaborative learning detailed in the USENIX Security 2021 proceedings. Cerebro addresses both the generality-performance tradeoff and the privacy-transparency challenge inherent in collaborative learning environments. It features a Python-like domain-specific language that allows users to write arbitrary learning programs, which are then optimized by an advanced compiler that selects the most efficient MPC protocols for execution. Additionally, Cerebro introduces innovative mechanisms such as compute policies and cryptographic auditing, enabling organizations to control the release of computed results and to audit the computation process without revealing private data. We will introduce this part in section 4.

Both platforms, CRYPTEN and Cerebro, are pivotal in advancing how sensitive data is utilized securely in machine learning. By facilitating collaborative learning without data exposure, these MPC implementations not only enhance data security, but also enable the realization of more sophisticated machine learning models. This review underscores the potential of MPC to revolutionize fields that require stringent data security measures, paving the way for widespread adoption in healthcare, finance, and beyond.

Central to the advancement of SecureML are specialized libraries designed to implement privacy-preserving techniques and cryptographic protocols seamlessly within ML workflows. In this paper, we focus on five prominent libraries: JustGarble, TFEncrypted, Crypten, LibOTe, and OTEExtension. These libraries, while diverse in their approaches, share common objectives and offer essential functionalities for enhancing privacy and security in ML applications. For this part, refer to section 5. We will also introduce the application of secure ML in the area of game development in section 6.

## 2 Secure ML: A System for Scalable Privacy-Preserving Machine Learning

### 2.1 Paper Abstract

SecureML presents a new and efficient approach to privacy-preserving machine learning, tackling the challenge of scalability in securely training models such as linear regression, logistic regression, and neural networks using stochastic gradient descent. The protocol operates under a two-server model where data owners distribute encrypted data across two non-colluding servers. These servers perform the computation without learning anything about the underlying data, using secure two-party computation (2PC).

### 2.2 Preliminaries

#### 2.2.1 Machine Learning

**Linear regression** Given  $n$  training data samples  $x_i$  each containing  $d$  features and the corresponding output labels  $y_i$ , regression is a statistical process to learn a function such that  $g(x_i) \approx y_i$ .

**Stochastic gradient descent (SGD)** SGD is an effective approximation algorithm for approaching a local minimum of a function, step by step. The SGD algorithm works as follows:  $w$  is initialized as a vector of random values or all 0s. In each iteration, a sample  $(x_i, y_i)$  is selected randomly and a coefficient  $w_j$  is updated as

$$w_j := w_j - \alpha \frac{\partial C_i(w)}{\partial w_j}$$

**Mini-batch** In practice, instead of selecting one sample of data per iteration, a small batch of samples are selected randomly and  $w$  is updated by averaging the partial derivatives of all samples on the current  $w$ .

**Learning rate adjustment** If the learning rate  $\alpha$  is too large, the result of SGD may diverge from the minimum. Therefore, a testing dataset is used to test the accuracy of current  $W$  and adjust the leaning rate  $\alpha$ .

**Termination** If the accuracy is lower than a specific threshold, we could terminate the learning process and output  $w$ .

**Logistic Regression** Logistic Regression is used to classify the dataset into two classes, the output label  $y$  is binary. Compared to linear regression, the logistic regression have an extra layer on which an activation function is applied on the result.

**Neural Network** Neural networks are a generalization of regression to learn more complicated relationships between high dimensional input and output data.

### 2.2.2 Secure Computation

**Oblivious Transfer** In an oblivious transfer protocol, a sender  $S$  has two inputs  $x_0$  and  $x_1$ , and a receiver  $R$  has a selection bit  $b$  and wants to obtain  $x_b$  without learning anything else or revealing  $b$  to  $S$ , which is denoted as

$$(\perp; x_b) \leftarrow OT(x_0, x_1; b)$$

The paper also use correlated OT extension [ALSZ13]. In the COT, the sender  $S$ 's input is related, which is  $x_0 = f(x_1)$ .

**Garbled Circuit 2PC** Garbled circuit could be used to securely compute any function by two parties. The paper require the garbling scheme to satisfy the standard security properties formalized in [BHR12].

**Secret Sharing and Multiplication Triplets** The paper use secret share to perform secure computation, which contains additively share  $Shr^A(\cdot)$ , reconstruction  $Rec^A(\cdot, \cdot)$  and multiply share  $Mul^A(\cdot, \cdot)$ . I want to go detail about the multiply share  $Mul^A(\cdot, \cdot)$ , which take advantage of Beaver's pre-computed multiplication triplet technique. The two parties have shared  $\langle u \rangle, \langle v \rangle, \langle z \rangle$ , where  $u, v \xleftarrow{\$} \mathbb{Z}_2$  and  $z = uv \bmod 2$ . Then  $P_i$  locally computes  $\langle e \rangle = \langle a \rangle - \langle u \rangle$  and  $\langle f \rangle = \langle b \rangle - \langle v \rangle$ . Both parties run  $Rec(\langle e_0 \rangle, \langle e_1 \rangle)$  and  $Rec(\langle f_0 \rangle, \langle f_1 \rangle)$  and  $P_i$  let  $\langle c_i \rangle = -i \cdot e \cdot f + f \cdot \langle a \rangle_i + e \cdot \langle b \rangle_i + \langle z \rangle_i$ . Note that Boolean sharing can be seen as additive sharing in  $\mathbb{Z}_2$ .

## 2.3 Secure Model

Let  $C_1, \dots, C_m$  denote the clients and  $S_0, S_1$  denote two non-colluding servers. We assume a semi-honest adversary  $A$  who can corrupt any subset of the clients and at most one of the two servers. The ideal functionality is described in figure 1.

<p>PARAMETERS: Clients <math>C_1, \dots, C_m</math> and servers <math>S_0, S_1</math>.</p> <p>UPLOADING DATA: On input <math>x_i</math> from <math>C_i</math>, store <math>x_i</math> internally.</p> <p>COMPUTATION: On input <math>f</math> from <math>S_0</math> or <math>S_1</math>, compute <math>(y_1, \dots, y_m) = f(x_1, \dots, x_m)</math> and send <math>y_i</math> to <math>C_i</math>. This step can be repeated multiple times with different functions.</p>
--

Figure 1: Ideal functionality

<p>THEOREM 1: In field <math>\mathbb{Z}_2</math>, let <math>x \in [0, 2^{l_x}] \cup [2^l - 2^{l_x}, 2^l]</math>, where <math>l &gt; l_x + 1</math> and given shares <math>\langle x \rangle_0, \langle x \rangle_1</math> of <math>x</math>, let <math>\langle \lfloor x \rfloor \rangle_0 = \lfloor \langle x \rangle_0 \rfloor</math> and <math>\langle \lfloor x \rfloor \rangle_1 = 2^l - \lfloor 2^l - \langle x \rangle_1 \rfloor</math>. Then with probability <math>1 - 2^{l_x + 1 - l}</math>, <math>Rec(\langle \lfloor x \rfloor \rangle_0, \langle \lfloor x \rfloor \rangle_1) \in \{\lfloor x \rfloor - 1, \lfloor x \rfloor, \lfloor x \rfloor + 1\}</math>, where <math>\lfloor \cdot \rfloor</math> denotes truncation by <math>l_D \leq l_x</math> bits.</p>
--

Figure 2: Truncation Theorem

## 2.4 Privacy Preserving Linear Regression

### 2.4.1 Vectorization

In order to benefit from the mini-batch and vectorization, the paper generalizes the addition and multiplication operations on share values to shared matrices.

### 2.4.2 Truncation

In order to improve the efficiency, the authors use a technique called truncation, which simply truncate the last half part of the decimal part of the recovered number and keep the front half part. The authors proved that this technique has a small influence on the accuracy of the final result. The theorem is described in Figure 2.

### 2.4.3 The online phase of privacy preserving linear regression

The online phase of privacy preserving linear regression is described in Figure 3. It assumed that the data-independent shared matrices  $\langle U \rangle, \langle V \rangle, \langle Z \rangle, \langle V' \rangle, \langle Z' \rangle$  were already generated in the offline phase. The protocol also requires multiplying the coefficient vector by  $\frac{\alpha}{|\bar{B}|}$  in each iteration, and  $\frac{\alpha}{|\bar{B}|}$  is set to be a power of 2, i.e.  $\frac{\alpha}{|\bar{B}|} = 2^{-k}$ . Then the multiplication with  $\frac{\alpha}{|\bar{B}|}$  can be replaced by having the parties truncate  $k$  additional bits from their shares of the coefficients.

### 2.4.4 The offline phase of privacy preserving linear regression

In the offline phase, the authors need to compute  $C = A \times B = \langle A \rangle_0 \times \langle B \rangle_0 + \langle A \rangle_0 \times \langle B \rangle_1 + \langle A \rangle_1 \times \langle B \rangle_0 + \langle A \rangle_1 \times \langle B \rangle_1$ . It is sufficient to compute  $\langle \langle A \rangle_0 \times \langle B \rangle_1 \rangle$  and  $\langle \langle A \rangle_1 \times \langle B \rangle_0 \rangle$  because the other two terms can be computed locally.

**Protocol**  $\text{SGD.Linear}(\langle \mathbf{X} \rangle, \langle \mathbf{Y} \rangle, \langle \mathbf{U} \rangle, \langle \mathbf{V} \rangle, \langle \mathbf{Z} \rangle, \langle \mathbf{V}' \rangle, \langle \mathbf{Z}' \rangle)$ :

- 1:  $\mathcal{S}_i$  computes  $\langle \mathbf{E} \rangle_i = \langle \mathbf{X} \rangle_i - \langle \mathbf{U} \rangle_i$  for  $i \in \{0, 1\}$ . Then parties run  $\text{Rec}(\langle \mathbf{E} \rangle_0, \langle \mathbf{E} \rangle_1)$  to obtain  $\mathbf{E}$ .
- 2: **for**  $j = 1, \dots, t$  **do**
- 3: Parties select the mini-batch  $\langle \mathbf{X}_{B_j} \rangle, \langle \mathbf{Y}_{B_j} \rangle$ .
- 4:  $\mathcal{S}_i$  computes  $\langle \mathbf{F}_j \rangle_i = \langle \mathbf{w} \rangle_i - \langle \mathbf{V}[j] \rangle$  for  $i \in \{0, 1\}$ . Then parties run  $\text{Rec}(\langle \mathbf{F}_j \rangle_0, \langle \mathbf{F}_j \rangle_1)$  to recover  $\mathbf{F}_j$ .
- 5:  $\mathcal{S}_i$  computes  $\langle \mathbf{Y}_{B_j}^* \rangle_i = -i \cdot \mathbf{E}_{B_j} \times \mathbf{F}_i + \langle \mathbf{X}_{B_j} \rangle_i \times \mathbf{F}_i + \mathbf{E}_{B_j} \times \langle \mathbf{w} \rangle_i + \langle \mathbf{Z}_j \rangle_i$  for  $i \in \{0, 1\}$ .
- 6:  $\mathcal{S}_i$  compute the difference  $\langle \mathbf{D}_{B_j} \rangle_i = \langle \mathbf{Y}_{B_j}^* \rangle_i - \langle \mathbf{Y}_{B_j} \rangle_i$  for  $i \in \{0, 1\}$ .
- 7:  $\mathcal{S}_i$  computes  $\langle \mathbf{F}'_j \rangle_i = \langle \mathbf{D}_{B_j} \rangle_i - \langle \mathbf{V}'_j \rangle_i$  for  $i \in \{0, 1\}$ . Parties then run  $\text{Rec}(\langle \mathbf{F}'_j \rangle_0, \langle \mathbf{F}'_j \rangle_1)$  to obtain  $\mathbf{F}'_j$ .
- 8:  $\mathcal{S}_i$  computes  $\langle \Delta \rangle_i = -i \cdot \mathbf{E}_{B_j}^T \times \mathbf{F}'_j + \langle \mathbf{X}_{B_j}^T \rangle_i \times \mathbf{F}'_j + \mathbf{E}_{B_j}^T \times \langle \mathbf{D}_{B_j} \rangle_i + \langle \mathbf{Z}'_j \rangle_i$  for  $i \in \{0, 1\}$ .
- 9:  $\mathcal{S}_i$  truncates its shares of  $\Delta$  element-wise to get  $\lfloor \langle \Delta \rangle_i \rfloor$ .
- 10:  $\mathcal{S}_i$  computes  $\langle \mathbf{w} \rangle_i := \langle \mathbf{w} \rangle_i - \frac{\alpha}{|B|} \lfloor \langle \Delta \rangle_i \rfloor$  for  $i \in \{0, 1\}$ .
- 11: Parties run  $\text{Rec}^A(\langle \mathbf{w} \rangle_0, \langle \mathbf{w} \rangle_1)$  and output  $\mathbf{w}$ .

Figure 3: The online phase of privacy preserving linear regression

**Protocol**  $\text{LHE.MT}(\langle \mathbf{A} \rangle_0; \langle \mathbf{B} \rangle_1)$ :

(Let  $a_{ij}$  be the  $(i, j)$ th element in  $\langle \mathbf{A} \rangle_0$  and  $b_j$  be the  $j$ th element in  $\langle \mathbf{B} \rangle_1$ .)

- 1:  $\mathcal{S}_1 \rightarrow \mathcal{S}_0$ :  $\text{Enc}(b_j)$  for  $i = 1, \dots, d$ .
- 2:  $\mathcal{S}_0 \rightarrow \mathcal{S}_1$ :  $c_i = \prod_{j=0}^d \text{Enc}(b_j)^{a_{ij}} \cdot \text{Enc}(r_i)$ , for  $i = 1, \dots, |B|$ .
- 3:  $\mathcal{S}_0$  sets  $\langle \langle \mathbf{A} \rangle_0 \times \langle \mathbf{B} \rangle_1 \rangle_0 = \mathbf{r}$ , where  $\mathbf{r} = (-r_1, \dots, -r_{|B|})^T \pmod{2^l}$ .
- 4:  $\mathcal{S}_1$  sets  $\langle \langle \mathbf{A} \rangle_0 \times \langle \mathbf{B} \rangle_1 \rangle_1 = (\text{Dec}(c_1), \dots, \text{Dec}(c_{|B|}))^T$ ,

Figure 4: The offline protocol based on linearly homomorphic encryption

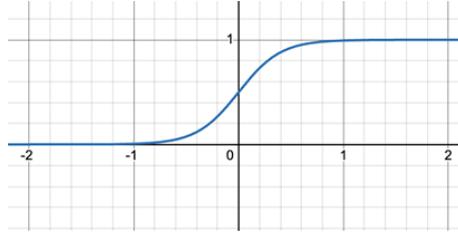


Figure 5: Sigmoid function

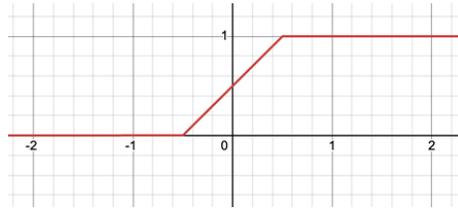


Figure 6: Secure computation friendly activation function

**LHE-based generation** The LHE-based generation involves the technique named linearly homomorphic encryption (LSE). The protocol can be found in Figure 4.

**OT-base generation** The shares of the product  $\langle A \rangle_0 \times \langle B \rangle_1$  could also be computed using OTs. The full protocol is a bit complex so that I will not go detail in this protocol. The full protocol is described detailed in [MZ17].

## 2.5 Privacy Preserving Logistic Regression

### 2.5.1 Secure computation friendly activation function

Because of the logistic activation Sigmoid function, described in Equation 1 and Figure 5, cannot be computed by multiplication and additive share, the author propose a new activation function that can be efficiently computed using secure computation techniques. The function is described in Equation 2 and Figure 6.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

$$f(x) = \begin{cases} 0 & x < -\frac{1}{2} \\ x + \frac{1}{2} & -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 1 & x \geq \frac{1}{2} \end{cases} \quad (2)$$

<p><b>Protocol</b> SGD.Logistic(<math>\langle \mathbf{X} \rangle, \langle \mathbf{Y} \rangle, \langle \mathbf{U} \rangle, \langle \mathbf{V} \rangle, \langle \mathbf{Z} \rangle, \langle \mathbf{V}' \rangle, \langle \mathbf{Z}' \rangle</math>):</p> <ol style="list-style-type: none"> <li>1: Do step 1–5 as in Figure 4. Both parties obtain the shares <math>\langle \mathbf{U}_{B_i} \rangle = \langle \mathbf{X}_{B_i} \times \mathbf{w} \rangle</math> (it was defined as <math>\langle \mathbf{Y}_{B_i}^* \rangle</math> in Figure 4).</li> <li>2: <b>for</b> every element <math>\langle u \rangle</math> in <math>\langle \mathbf{U}_{B_i} \rangle</math> <b>do</b></li> <li>3: <math>(\langle b_3 \rangle^B, \langle b_4 \rangle^B) \leftarrow \text{Y2B}(\text{GarbledCircuit}(\langle u \rangle_0 + \frac{1}{2}, \langle u \rangle_0 - \frac{1}{2}; \langle u \rangle_1, f))</math>, where <math>f</math> sets <math>b_1</math> as the most significant bit of <math>(\langle u \rangle_0 + \frac{1}{2}) + \langle u \rangle_1</math> and <math>b_2</math> as the most significant bit of <math>(\langle u \rangle_0 - \frac{1}{2}) + \langle u \rangle_1</math>. It then outputs <math>b_3 = \neg b_1</math> and <math>b_4 = b_1 \wedge (\neg b_2)</math>.</li> <li>4: <math>\mathcal{S}_0</math> sets <math>m_0 = \langle b_4 \rangle_0^B \cdot \langle u \rangle_0 + r_1</math> and <math>m_1 = (1 - \langle b_4 \rangle_0^B) \cdot \langle u \rangle_0 + r_1</math>. <math>\mathcal{S}_0</math> and <math>\mathcal{S}_1</math> run <math>(\perp; m_{\langle b_4 \rangle_1^B}) \leftarrow \text{OT}(m_0, m_1; \langle b_4 \rangle_1^B)</math>. <math>m_{\langle b_4 \rangle_1^B}</math> is equal to <math>(\langle b_4 \rangle_0^B \oplus \langle b_4 \rangle_1^B) \cdot \langle u \rangle_0 + r_1 = b_4 \cdot \langle u \rangle_0 + r_1</math>.</li> <li>5: <math>\mathcal{P}_1</math> sets <math>m_0 = \langle b_4 \rangle_1^B \cdot \langle u \rangle_1 + r_2</math> and <math>m_1 = (1 - \langle b_4 \rangle_1^B) \cdot \langle u \rangle_1 + r_2</math>. <math>\mathcal{S}_1</math> and <math>\mathcal{S}_0</math> run <math>(\perp; m_{\langle b_4 \rangle_0^B}) \leftarrow \text{OT}(m_0, m_1; \langle b_4 \rangle_0^B)</math>. <math>m_{\langle b_4 \rangle_0^B}</math> is equal to <math>b_4 \cdot \langle u \rangle_1 + r_2</math>.</li> <li>6: <math>\mathcal{S}_0</math> sets <math>m_0 = \langle b_3 \rangle_0^B + r_3</math> and <math>m_1 = (1 - \langle b_3 \rangle_0^B) + r_3</math>. <math>\mathcal{S}_0</math> and <math>\mathcal{S}_1</math> run <math>(\perp; m_{\langle b_3 \rangle_1^B}) \leftarrow \text{OT}(m_0, m_1; \langle b_3 \rangle_1^B)</math>. <math>m_{\langle b_3 \rangle_1^B}</math> is equivalent to <math>b_3 + r_3</math>.</li> <li>7: <math>\mathcal{S}_0</math> sets <math>\langle y^* \rangle_0 = m_{\langle b_4 \rangle_0^B} - r_1 - r_3</math> and <math>\mathcal{S}_1</math> sets <math>\langle y^* \rangle_1 = m_{\langle b_4 \rangle_1^B} + m_{\langle b_3 \rangle_1^B} - r_2</math>.</li> <li>8: <b>end for</b></li> <li>9: Both parties set <math>\langle \mathbf{Y}^* \rangle_i</math> as a vector of all <math>\langle y^* \rangle_i</math>s computed above and continue to step 6–12 in Figure 4.</li> </ol>
--

Figure 7: Privacy preserving logistic regression protocol

## 2.5.2 Privacy preserving logistic regression protocol

Firstly, we need to define two variables as described in Equation 3 and Equation 4. Use these two variables, we could update the activation function in Equation 2 to

$$f(u) = (\neg b_2) + (b_2 \wedge (\neg b_1))u$$

Use this activation function, the full protocol of Privacy preserving logistic regression is described in Figure 7.

$$b_1 = \begin{cases} 0 & u + \frac{1}{2} \geq 0 \\ 1 & u + \frac{1}{2} < 0 \end{cases} \quad (3)$$

$$b_2 = \begin{cases} 0 & u - \frac{1}{2} \geq 0 \\ 1 & u - \frac{1}{2} < 0 \end{cases} \quad (4)$$

# 3 CRYPTEN: Secure Multi-Party Computation Meets Machine Learning

## 3.1 Introduction

CRYPTEN introduces a novel software framework designed to bridge the gap between secure multi-party computation (MPC) and machine learning. Developed by Facebook AI Research, it aims to make secure MPC techniques accessible and practical for machine learning researchers and engineers who may not have a deep background in cryptography.

## 3.2 Framework Design

CRYPTEN is structured around a PyTorch-like API, offering familiar machine learning abstractions, such as tensors, automatic differentiation, and neural networks, all designed to operate within a secure MPC environment. This design allows machine learning practitioners to leverage secure MPC methods without needing to understand the underlying cryptographic details.

## 3.3 GPU Support and Scalability

A distinguishing feature of CRYPTEN is its support for GPU computations, which significantly enhances the performance of MPC operations, allowing for more complex and larger scale machine learning tasks to be conducted securely and efficiently.

## 3.4 Flexible Use Case Applications

The framework is demonstrated to be capable of supporting a wide range of machine learning tasks from text classification to speech recognition and image classification, providing a robust tool for privacy-preserving machine learning.

# 4 Cerebo - A Real-World implementation

## 4.1 Platform Overview

### 4.1.1 Threat model

After discussing the underlying building-blocks of Secure-ML, we will go through a recent platform that allows real-world users to perform those ideal functionalities with no prior knowledge about MPC or Secure-ML. The platform named Cerebo supports  $P$ -parties to perform a single learning task with their secret datasets. It also allows users to choose between two threat modes: semi-honest and malicious settings. Both settings are defined in the context of cryptography. In other words, the model can only prevent information leakage of users' datasets when  $n - 1$  parties are corrupted and form a semi-honest adversary or there are parties deviate from the protocol. In the high level, how Cerebo achieves both is by pre-defined sub-protocols and inconsistent data which will appear when misbehavior happening. However, in machine learning world, instead of the protocol, the trained model could also be easily attacked by carefully constructed dataset. An interesting example is that a skin cancer detection model<sup>1</sup> had mistakenly thought every image that contained ruler marking was indicative of melanoma. However, this was only because of the ruler markings contained in most of the images of malignant lesions. Though this example is easy to detect when training is performed publicly, under MPC settings, the

---

<sup>1</sup>Problematic cancer detection model <https://www.sciencedirect.com/science/article/pii/S0022202X18322930>

definition of MPC does not allow anyone to learn any information about the training set. Therefore, Cerebro cannot perform any work related with identifying maliciously constructed datasets even if the attack method is as naive as the example above [LVK<sup>+</sup>24].

### 4.1.2 Implementation

Cerebro is itself implemented by python and another MPC platform named SCALE-MAMBA <sup>2</sup> which contains extensive pre-implemented MPC protocols. To use Cerebro, users are assumed to know basic python grammar and some specific instructions, and the platform will automatically compile the instruction and run them in the correct mode. The way Cerebro achieves such functionality is by the use of Domain Specific Language(DSL), which guarantees the privacy level of variables after they are created. During compilation phase, compiler will choose the running mode accordingly by observing the input data types. If at least one private variable shows up in the context, the platform will switch into secure mode and perform everything (from basic data exchanging to matrix addition and multiplication) by the predefined protocols. Moreover, the platform also makes a copy for for the original instructions and store it locally. The local version is used for local computation, which will be further discussed in next section. Now, the online version is generated and the running mode is determined, the compiler performs another technique called fused operator which will also be discussed later. Now, the online version of the task is ready, and before the beginning of the task, Cerebro first does pre-processes which prepares random masks and generates key pairs. In the mean time, the local machines also generate commitments on their dataset and check the commitments received from others. Once everything is done, the well-formed online training task starts to run, and the result model is published to every parties. Notice that the final model should be transparent to every party, and that is everyone should know every single parameters of the final model. This is one of the main reasons why parties could ask Cerebro to enter auditing mode.

## 4.2 Optimization

### 4.2.1 Compiler

The first optimization of Cerebro is pre-defined behavior of the compiler. To speed up the computation, the compiler can choose the runtime mode, perform fused operator, and determine the pre-processing mode. Note that we've discussed the runtime mode decision above. Before explaining the remaining ones, it's also worth to mentioned that the local plain-text version of code can further improve the performance simply because if we want to use the sum of two secret values, Cerebro could add them together locally and then encrypt the result instead of directly going into online phase and doing anything secretly. Furthermore, fused operator[PTN<sup>+</sup>18] is commonly used in program compiler,

---

<sup>2</sup>SCALE-MAMBA <https://github.com/KULeuven-COSIC/SCALE-MAMBA>

enabling the compiler to find operations that could be fused into one to improve performance.

#### 4.2.2 Pre-processing Mode

After the fusing operation, the final version is submitted to the server, and the pre-processing strategy will be selected from two pre-defined ones. The major computation cost for both of them is to generate random triplets for the matrix multiplication. From the Secure-ML paper, we know that the Beaver’s triplets trick can also be applied to matrix multiplication by vectorization. Therefore, for each epoch, we need a random triplets chart to mask all multiplication happens between an  $m \times n$  sample matrix and an  $n \times 1$  weights vector. The paper defines the costs of two strategies very formally since Cerebro makes decisions based on the magnitude of results of the two costs. The strategies is called linear pre-processing and quadratic pre-processing. At a high level, the first protocol ask all parties to generate random data for charts used in all epochs at once, parties will encrypt their result and send all of them to the server. Then, the server will generate  $P$  long lists of charts based on what it received and publish the result to each party. Define the cost of encrypting  $n$  bits to be  $\mathbf{EncCost}(n)$ , let one mask chart to be  $m$  bits long, and assume the learning task takes  $e$  epoches, so  $e$  charts are needed in total. Then the final cost for one party is roughly:

$$\mathbf{EncCost}(e \times m) + O(1) \text{ communication}$$

Since each party needs to encrypt all data at once, and sending & receiving data from sever only takes constant round.

Using the same setting, the quadratic pre-processing strategy asks each party to communicate with all others for constant round to form part of the random mask for one epoch. Parties will then send all encrypted charts to the server, and server will generate the final part of the chart. Now, the cost is lowered for each encryption, but the communication cost arise faster. The cost for this strategy for one party is roughly:

$$e \times \mathbf{EncCost}(m) + O(P) \text{ communication}$$

Where  $P$  is the number of parties. The sever will evaluate both costs and choose the cheaper strategy to perform.

#### 4.2.3 Hierarchical layout

The paper also discusses a typical scenario where various parties are interested in collaboratively calculating the sum of their data using homomorphic encryption with varying bandwidth capacities. This problem is particularly relevant when considering real-world scenario, such as the lower bandwidth typically available between different continents compared to that between regions within the same

country. To address this disparity and optimize the use of limited bandwidth, the paper proposes the concept of a local aggregator. Each local aggregator is responsible for summing the data within its vicinity and then only utilizes the constrained, narrow bandwidth routes for communicating with other aggregators. This strategy significantly reduces the reliance on these slower connections. By implementing homomorphic encryption, which allows computations to be performed on encrypted data without needing to decrypt it, the system can minimize the frequency of data transmission over these narrow routes. The number of times data passing through such routes can be reduced to a logarithmic scale in the most efficient scenarios. This approach enhances the overall efficiency and feasibility of distributed computing tasks across geographically dispersed parties.

### **4.3 Auditing**

#### **4.3.1 Cross Validation**

To deal with malicious input data, the paper suggests two auditing methods. The first is Cross Validation, which chooses each party as suspicion at a time, remove its dataset, and retrain the whole model. If there is a round that the model performs significantly better, we mark the suspicion in that round as cheater. However, this method is extremely inefficient because once the number of suspicions grows up, we will need to retrain the model for factorial number of times, which is completely impractical. Moreover, there is another attack method called backdoor injection, which is another type of maliciously constructed training data that allows the attacker to control the behavior of the model. Such dataset is impossible to be detected by cross validation since it merely affect the performance.

#### **4.3.2 Commit on input**

The paper proposes a commitment scheme also designed to counter data poisoning when parties collaborating in a distributed computation environment. Initially, each party commits to their data inputs by sharing a cryptographic commitment with the other parties involved. This commitment serves as a secure record of their original data, preserved for future verification. After the training phase, if any party initiates an audit process (when abnormal behavior is detected or the model perform worse than the case that it was only trained with a party’s own dataset), all parties are required to resubmit their datasets for verification. At this stage, Cerebro performs a consistency check to ensure that the data provided both before and after the learning task remains unchanged, confirming that no tampering has occurred during the computation process.

However, there is a potential vulnerability inherent in MPC protocols used in this scheme. While the consistency check ensures data integrity without revealing the actual contents of the inputs and thus maintaining confidentiality,

it cannot fully prevent dishonest behavior. In scenarios where a party claims loss of the original data, the system struggles to enforce accountability conclusively. This is because, despite the initial commitments, MPC does not allow for the recovery or reconstruction of the original data from the protocol alone. Therefore, if a party were to falsely claim that their data was lost, other parties would have no definitive means to dispute this claim or prove the dishonesty, which could be one of the major security issues of Cerebro.

#### 4.4 Experiment & Evaluation

The paper provides a comprehensive analysis of various design elements and optimizations that significantly enhance performance across different computational scenarios, especially when evaluated using a detailed cost model. These optimizations are shown to dramatically improve system efficiency under conditions tailored to their specific strengths.

However, there are notable limitations in the implementation of certain features. Specifically, in the cost model part, all parties are asked to send their encrypted data for random mask generation. Though the paper previously states that Cerebro can work in maliciously secure mode, their cost model implicitly assumes that the server works like a trusted-third-party, which might not always be the case in reality. Though the paper does not provide much of technical detail about the generation of random mask, I assume they are using multiplicative homomorphic encryption to encrypt those triplets with commitment scheme. If so, a malicious server could manipulate the input data to generate malicious outputs. More discussions might be needed to explain their data encryption and manipulation schemes.

Moreover, when conducting secure logistic regression with strong security guarantees against malicious actors, the system encounters memory exhaustion. This issue indicates a need for further refinement in their approach to handling large-scale secure computations, suggesting that the method may require additional optimization to reduce memory usage effectively.

Another critical area of concern is the commitment scheme used to ensure data integrity and non-repudiation among the participating parties. The current protocol requires each party to create a commitment by first generating a hash of their data, committing on the hash, and then signing this commitment. This method, when applied without batching, is highly inefficient; the process takes an extended period of 4.5 days to verify commitments for 27,000 records. In contrast, when a batching-commitment approach is employed, the verification time is drastically reduced to approximately 2.23 hours, which is reasonable but still inefficient.

Furthermore, the batching-commitment scheme also introduces a new problem: it requires the data order to remain unchanged. Any alteration in the sequence of the records leads to inconsistencies that the system incorrectly flags as data tampering (false positives). This limitation restricts the flexibility of the system in dynamic environments where data order might change for legitimate reasons. This problem reveals another potential area for improvement

in ensuring the scheme’s robustness and adaptability without compromising on security.

## 5 Libraries and their implementation for Secure ML

### 5.1 Introduction

Machine learning (ML) has the potential to revolutionize industries from healthcare to finance. Yet, the sensitive data at the heart of these domains often presents a formidable barrier to innovation. Secure ML libraries emerge as a game-changing solution, enabling organizations to leverage the power of ML while safeguarding the confidentiality and integrity of their most valuable assets.

These libraries encapsulate a wealth of cryptographic techniques and secure protocols, allowing stakeholders to collaboratively train and deploy ML models without exposing their underlying data. From the robust collaborative frameworks of secure multi-party computation (MPC) to the powerful encryption techniques of homomorphic encryption (HE) and the nuanced privacy guarantees of differential privacy (DP), Secure ML libraries offer a comprehensive toolkit for balancing data utility and privacy preservation.

#### 5.1.1 JustGarble

JustGarble emerges as a powerful tool in the realm of secure multi-party computation (MPC) and oblivious transfers (OT). Its streamlined approach to circuit garbling and evaluation simplifies the process, making it more accessible and efficient. With JustGarble, you can easily transform circuits into the Simple Circuit Description (SCD) format using external compilers.[BHKR13]

##### **Circuit Representation:**

- Let  $C$  be the Boolean circuit representing the function to be computed.
- $C$  consists of gates  $g_1, g_2, \dots, g_n$  and wires connecting these gates.
- Each gate  $g_i$  has input wires  $w_{i1}, w_{i2}, \dots, w_{ik}$  and output wire  $w_i$ .

**Garbling Phase:** In SecureML, User 1 (Alice) acts as the garbler. Here’s how it works:

- For each gate  $g_i$  in the Boolean circuit, Alice assigns two random labels (garbled labels) to each wire of the gate, representing the output of the gate for inputs 0 and 1.
- These labels, denoted as  $L_{i,j,0}$  and  $L_{i,j,1}$ , correspond to the  $j$ -th wire of gate  $g_i$  for inputs 0 and 1, respectively.

- Alice encrypts these garbled labels using symmetric encryption with keys derived from the input values.
- For example,  $E_k(L_{i,j,b})$  represents the encrypted label for wire  $j$  of gate  $g_i$  with input  $b$ .
- Additionally, Alice generates and encrypts input labels for her inputs.
- Alice then sends the encrypted garbled circuit to User 2 (Bob).

**Evaluation Phase:** In SecureML, User 2 (Bob) acts as the evaluator. Here's how it works:

- Bob obtains his input labels by using oblivious transfer (OT) to select the appropriate encrypted garbled labels from Alice's transmissions.
- For each wire corresponding to his input, Bob uses OT to receive the encrypted garbled labels from Alice.
- Bob proceeds gate by gate, evaluating each gate using the garbled labels he received from Alice.
- For each gate, Bob decrypts the garbled labels corresponding to the input wires using the keys derived from his input values.
- Based on the truth table of the gate, Bob reveals the garbled label of the output wire.
- Bob sends the garbled label for the output wire to Alice.
- Alice decrypts the received garbled label to obtain the final output of the function.

SecureML enables two parties to work together, harnessing the power of their combined data without ever revealing their individual inputs. It's like a magic trick where both parties contribute ingredients to a recipe, but neither can see the other's secret ingredients until the dish is complete. SecureML achieves this through a clever mix of symmetric encryption and oblivious transfer, ensuring that communication and computation remain secure and private at every step.

### 5.1.2 TFEncrypted

TFEncrypted opens up new possibilities for collaborative machine learning. It enables multiple parties to work together, leveraging the power of their combined data without ever revealing their individual inputs. It's like a magic trick where everyone contributes to the illusion, but the secrets remain hidden until the grand reveal.

This is made possible by TFEncrypted's ability to automatically convert TensorFlow computation graphs into secure computation graphs using advanced

MPC protocols.[Dev] It's a powerful tool for organizations looking to harness the potential of distributed data without compromising privacy. TFEncrypted facilitates the following key steps in the process:

1. **Defining Computation Graphs:** Users define TensorFlow computation graphs specifying operations such as matrix multiplications, convolutions, and activations.
2. **Securing the Graph:** TFEncrypted converts the computation graph into a secure computation graph using MPC protocols. This involves partitioning the computation among multiple parties, encrypting inputs, and performing secure computations on encrypted data.
3. **Execution:** The secured computation graph can be executed by multiple parties, with each party holding encrypted shares of the input data. Through secure computations, parties collaborate to perform the desired computation while ensuring privacy and security.

TFEncrypted relies on Secure MPC protocols to ensure privacy and security in collaborative machine learning tasks. Some key concepts involved in Secure MPC include:

- **Secret Sharing:** Inputs are divided into shares using secret sharing schemes such as Shamir's Secret Sharing, ensuring that no individual party can reconstruct the original inputs.
- **Secure Operations:** Computations are performed on shares of inputs using cryptographic protocols, preserving privacy and correctness. These protocols involve interactions between parties to compute functions while keeping inputs and intermediate values private.
- **Consistency:** At the end of the computation, parties collaborate to reconstruct the final result using the shares they collectively hold, ensuring consistency with the result of performing the computation on plaintext inputs.

### 5.1.3 Crypten

Crypten is pushing the boundaries of what's possible in secure machine learning. This groundbreaking framework leverages secure multi-party computation (MPC) to automatically convert standard machine learning computations into operations on encrypted data. This enables multiple parties to collaborate on machine learning tasks without revealing their private data.

In this paper, we provide an overview of Crypten, explore the cryptographic concepts it utilizes, and delve into the mathematical foundations that make it possible. Crypten opens up new avenues for leveraging distributed data without compromising privacy, unlocking the full potential of collaborative machine learning. The framework facilitates the following key steps in the process:

1. **Defining Computation Graphs:** Users define computation graphs using familiar deep learning frameworks such as PyTorch. These graphs specify operations such as neural network layers, activations, and loss functions.
2. **Encryption:** Crypten automatically encrypts the computation graph and input data using homomorphic encryption techniques. This allows computations to be performed directly on encrypted data while preserving privacy.
3. **Secure Evaluation:** The encrypted computation graph is securely evaluated using secure MPC protocols. Parties collaborate to perform computations on encrypted data without revealing their inputs, ensuring privacy and security.

Crypten relies on Secure MPC protocols to ensure privacy and security in machine learning tasks. Key concepts involved in Secure MPC include:

- **Homomorphic Encryption:** Crypten employs homomorphic encryption schemes to enable computations on encrypted data. These schemes allow operations such as addition and multiplication to be performed directly on ciphertexts, preserving the privacy of the underlying data.
- **Secure Aggregation:** Secure MPC protocols enable parties to securely aggregate their encrypted shares to compute functions without revealing their inputs. This ensures that computations can be performed collaboratively while maintaining privacy.
- **Zero-Knowledge Proofs:** Crypten utilizes zero-knowledge proofs to validate computations performed by parties without revealing any information about the inputs or intermediate values. This provides an additional layer of security against malicious parties.

#### 5.1.4 LibOTe

LibOTe is pushing the boundaries of what’s possible in secure communication. This groundbreaking library leverages Oblivious Transfer (OT) techniques to automatically convert standard communication protocols into secure exchanges of encrypted data.[Gro] This enables parties to collaborate without revealing their private data.

In this paper, we provide an overview of LibOTe, explore the cryptographic concepts it utilizes, and delve into the mathematical foundations that make it possible. LibOTe opens up new avenues for secure communication, unlocking the full potential of collaborative information exchange. The library facilitates the following key functionalities:

1. **Oblivious Transfer (OT):** LibOTe offers efficient implementations of various OT protocols, allowing parties to securely transfer information while preserving privacy.

2. **Secure Channels:** The library provides mechanisms for establishing secure channels between parties, ensuring confidentiality and integrity of communications.
3. **Error Correction:** LibOTe includes error correction techniques to enhance the reliability of communication over insecure channels.

LibOTe relies on secure communication protocols to ensure privacy and security in data exchange. Key concepts involved in secure communication include:

- **Oblivious Transfer (OT):** OT protocols enable one party (the sender) to securely transfer information to another party (the receiver) without learning which information was sent. This ensures that the sender's privacy is preserved.
- **Secure Channels:** Secure channels provide a means for parties to communicate securely over untrusted networks. These channels typically involve encryption, authentication, and integrity verification mechanisms to protect against eavesdropping and tampering.
- **Error Correction:** Error correction techniques are employed to detect and correct errors introduced during data transmission. This ensures the reliability of communication, even in the presence of noise and interference.

LibOTe involves advanced cryptographic primitives and communication protocols. These include:

- **OT Protocols:** Various OT protocols, such as 1-out-of-2 OT and k-out-of-n OT, are based on mathematical concepts such as number theory, algebra, and probability theory. These protocols ensure that the sender's privacy is preserved while enabling secure data transfer.
- **Error Correction Codes:** Error correction codes, such as Hamming codes and Reed-Solomon codes, are based on linear algebra and coding theory. These codes enable the detection and correction of errors introduced during data transmission, enhancing the reliability of communication.
- **Cryptography:** Cryptographic primitives, such as symmetric and asymmetric encryption, digital signatures, and hash functions, are fundamental to secure communication protocols. These primitives ensure confidentiality, integrity, and authenticity of data exchanged between parties.

LibOTe offers a versatile library for implementing secure communication protocols, with a focus on Oblivious Transfer techniques. By leveraging advanced cryptographic primitives and communication protocols, LibOTe enables parties to exchange information securely while preserving privacy and integrity.

### 5.1.5 OTEExtension

OTEExtension is pushing the boundaries of what's possible in secure computation. This groundbreaking protocol leverages Oblivious Transfer (OT) techniques to efficiently perform secure one-out-of-two operations over the internet. This enables parties to collaborate without revealing their private data.[PSZ18]

In this paper, we provide an overview of OTEExtension, explore the cryptographic concepts it utilizes, and delve into the mathematical foundations that make it possible. OTEExtension opens up new avenues for secure computation, unlocking the full potential of collaborative information exchange. The protocol facilitates the following key functionalities:

1. **Efficient OT:** OTEExtension enables parties to perform one-out-of-two oblivious transfer efficiently, even over high-latency and low-bandwidth networks.
2. **Secure Computation:** The protocol ensures that both the sender and receiver of OT messages remain oblivious to each other's inputs, preserving privacy and security.
3. **Scalability:** OTEExtension is designed to scale to a large number of OT instances, making it suitable for applications requiring a high volume of secure computations.

OTEExtension relies on oblivious transfer protocols to achieve secure communication between parties. Key concepts involved in oblivious transfer include:

- **OT Protocols:** OT protocols enable one party (the sender) to transfer a message to another party (the receiver) without learning which message was sent. This ensures that the receiver's privacy is preserved.
- **Security Guarantees:** OT protocols provide security guarantees such as sender privacy, receiver privacy, and correctness. These guarantees are achieved through cryptographic techniques such as encryption, commitment schemes, and zero-knowledge proofs.
- **Efficiency:** Efficient OT protocols, such as OTEExtension, optimize communication and computation costs to enable secure communication over the internet.

OTEExtension involves advanced cryptographic primitives and protocols. These include:

- **Commitment Schemes:** Commitment schemes are cryptographic protocols that allow a party to commit to a value without revealing it. These schemes are used in OTEExtension to ensure that both parties remain oblivious to each other's inputs until the OT is completed.

- **Zero-Knowledge Proofs:** Zero-knowledge proofs are cryptographic techniques that allow a party to prove knowledge of a value without revealing the value itself. These proofs are used in OTExtension to verify the correctness of OT instances without revealing the chosen messages.
- **Efficient Protocols:** OTExtension utilizes efficient cryptographic protocols and optimizations to minimize communication and computation costs, enabling scalable and secure OT over the internet.

OTExtension offers robust security for oblivious transfer over the internet. This powerful protocol leverages advanced cryptographic techniques to ensure parties can exchange sensitive information securely and efficiently. With OTExtension, organizations can collaborate with confidence knowing their data remains protected and its integrity preserved.

## 5.2 Multiparty Computation

These libraries for Secure MPC involves advanced cryptographic primitives such as homomorphic encryption, secret sharing schemes, and zero-knowledge proofs. These primitives enable computations to be performed on encrypted data while preserving privacy and correctness. For example, homomorphic encryption allows operations to be performed directly on ciphertexts, enabling secure computation on encrypted data without decryption.

# 6 Secure ML in games

## 6.1 Introduction

In the world of adversarial games, where strategic decision-making and information concealment are key, Secure ML techniques powered by frameworks like TFEncrypted offer a transformative approach to conducting analyses while safeguarding sensitive information.

Traditionally, adversarial games [GWHX21] involve multiple players making sequential moves with imperfect information, often leading to complex decision trees and strategic interactions. However, in scenarios where players are distributed geographically or are unwilling to reveal their private strategies, conventional methods fall short due to privacy concerns. Secure ML techniques address these challenges by enabling collaborative learning without the need to share raw data or strategies.

By leveraging homomorphic encryption and secure aggregation techniques, TFEncrypted allows players to train models and make strategic decisions collaboratively while keeping their individual strategies and preferences confidential. This opens up new possibilities for secure, distributed adversarial game analysis.

In this paper, we explore the application of TFEncrypted in the context of adversarial games, focusing on two prominent examples: Tic Tac Toe and Poker. We delve into the methodologies employed to secure the gameplay, including

the encryption of game states and the execution of strategic algorithms within a secure computation environment. Furthermore, we discuss the challenges and opportunities presented by Secure ML in adversarial settings, highlighting the potential for advancing strategic decision-making while preserving player privacy.

## 6.2 Secure ML in TicTacToe

Conventional ML algorithms typically require access to raw game data, including past game states and player actions, to train predictive models and strategic algorithms. However, sharing such data among players in a multiplayer setting raises concerns regarding privacy and data security. Without proper safeguards in place, players may be reluctant to disclose their game history and strategies, leading to asymmetries in the availability of information and unfair gameplay dynamics. Moreover, traditional ML methods often lack mechanisms to ensure the integrity and confidentiality of strategic decision-making processes. In Tic Tac Toe[Pau20], for instance, players may exploit vulnerabilities in the learning algorithms to infer opponents’ strategies or manipulate game outcomes, compromising the fairness and competitiveness of the game.

These shortcomings inherent in conventional machine learning when applied to Tic Tac Toe highlight the necessity for implementing Secure ML practices including TFEncrypted. Such techniques are designed to confront issues surrounding privacy, justice, and cooperative endeavors within adversarial gaming realms. Secure ML, fortified with cryptographic protocols and encryption methods, empowers participants to collaboratively partake in strategic thinking while concurrently protecting individual privacy and maintaining the integrity of their gameplay interactions. This research showcases the successful deployment of TFEncrypted within a Tic Tac Toe landscape as evidence of Secure ML’s potential to elevate strategy formulation processes and uphold fairness within multi-player gaming contexts.

Q-learning is a reinforcement learning algorithm that learns optimal strategies through trial and error. At its core, Q-learning involves updating a Q-value table based on rewards received from different game states. The Q-value represents the expected cumulative reward when taking a particular action from a given state. The update rule for Q-learning can be defined as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot \left( r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a) \right)$$

where:

- $Q(s, a)$  is the Q-value for state  $s$  and action  $a$ .
- $r$  is the immediate reward received after taking action  $a$  from state  $s$ .
- $\alpha$  is the learning rate (a parameter controlling the rate of learning).
- $\gamma$  is the discount factor (a parameter balancing immediate and future rewards).

- $s'$  is the next state after taking action  $a$ .
- $a'$  is the next action chosen based on the maximum Q-value in the next state  $s'$ .

**Using TFEncrypted :** To implement Q-learning in Tic Tac Toe using TFEncrypted, we can follow these steps:

1. **Game Representation:** Represent the Tic Tac Toe game state as a vector of length 9, where each element corresponds to a cell on the board (0 for empty, 1 for X, -1 for O).
2. **Model Initialization:** Initialize a Q-value table as a TensorFlow variable, encrypted using TFEncrypted's secure computation protocols. Each entry in the table represents the Q-value for a specific game state-action pair.
3. **Training Loop:** Iterate through episodes of gameplay, where the AI agent interacts with the environment (the Tic Tac Toe board). At each step, the agent selects an action based on an exploration-exploitation strategy (e.g., epsilon-greedy) and updates the Q-value table using the Q-learning update rule. The AI agent's actions and rewards are encrypted using TFEncrypted to preserve privacy.
4. **Secure Evaluation:** During gameplay, both the AI agent and the human opponent interact with the encrypted Q-value table using TFEncrypted's secure computation protocols. This ensures that the AI agent's strategy and decision-making process remain private from the opponent, enhancing fairness and preserving player privacy.
5. **Deployment:** Once the AI agent is trained, deploy it to play against human opponents in a secure and privacy-preserving manner. TFEncrypted ensures that the AI agent's strategy and decision-making process remain confidential during gameplay, fostering a fair and competitive gaming environment.

Through the integration of TFEncrypted in Tic Tac Toe with ML, this approach enables players to engage in strategic gameplay while preserving privacy and ensuring fairness. By leveraging secure computation protocols and encryption mechanisms, TFEncrypted facilitates collaborative learning and decision-making in adversarial gaming scenarios, paving the way for enhanced player experiences and privacy-preserving gaming environments.

### 6.3 Using Decision Trees in Poker with TFEncrypted

Decision trees stand out as a robust algorithm within machine learning, championing both classification and regression challenges. When applied to the realm of Poker, these analytical models delve into the heart of strategy, parsing through

the current dynamics of the game and the psychological nuances of player tactics. By iteratively partitioning the feature space, decision trees meticulously carve out subsets in alignment with attribute values until reaching the terminal leaves where definitive gameplay choices are rendered. This logical dissection is visually encapsulated in a dendritic diagram, where each node maps to an influential factor, and the branches symbolize the potential strategic forks in the road. To implement decision trees in Poker using TFEncrypted[BPTG15], we can follow these steps:

1. **Game Representation:** Represent the Poker game state and player actions as features in a dataset. Each row in the dataset corresponds to a unique game state, while columns represent different features such as current hand strength, bet size, and opponent behavior.
2. **Model Training:** Train a decision tree classifier on the encrypted dataset using TFEncrypted's secure computation protocols. The decision tree learns to predict the optimal action to take in a given game state based on the features provided.
3. **Secure Evaluation:** During gameplay, both the AI player and human opponents interact with the decision tree model using TFEncrypted's secure computation protocols. The AI player's actions and observations are encrypted to ensure privacy, while the decision tree model remains confidential.
4. **Strategic Decision-Making:** The decision tree model guides the AI player in making strategic decisions during Poker gameplay. Based on the current game state and observed features, the AI player consults the decision tree to determine the optimal action to take, such as folding, calling, or raising.
5. **Fair and Competitive Gameplay:** TFEncrypted ensures that both the AI player and human opponents can engage in Poker gameplay in a fair and competitive manner, with their strategies and decision-making processes kept private. This fosters a level playing field and enhances the overall gaming experience.

By incorporating TFEncrypted into the realm of Poker utilizing decision trees, this method empowers participants to engage in superior strategic gameplay under the banner of privacy protection and equitable play. TFEncrypted harnesses the power of cutting-edge secure computation protocols together with sophisticated encryption techniques, thus catalyzing a collaborative learning and decision-making revolution in the often contentious world of gaming. The result is a groundbreaking leap forward for player engagement and the establishment of gaming enclaves where privacy is not just a feature, but a fundamental standard.

## 7 MP-SPDZ: A Versatile Framework for Multi-Party Computation

### 7.1 Introduction

Multi-Party Computation (MPC) protocols facilitate collaborative computation amongst several entities on a function, hinging on their confidential inputs. This collaborative effort is conjoined with a commitment to input privacy. These protocols are pivotal for the integrity and confidentiality in environments where the joint processing of sensitive information is mandatory. MP-SPDZ emerges as a flexible and dynamic framework pivotal for MPC that furnishes robust, scalable solutions tailored for a spectrum of applications that necessitate the preservation of privacy. [KD20]

### 7.2 Background

Multi-Party Computation (MPC) has risen as an indispensable mechanism for fortifying privacy and ensuring security within the realm of collaborative computing engagements. At its core, MPC protocols are designed to allow multiple entities to collaboratively execute a function while maintaining the confidentiality of their individual inputs. Nevertheless, the deployment of MPC protocols in a manner that is both efficient and secure remains a formidable challenge, particularly in environments characterized by an extensive number of participants or where the computational endeavors are intricate.

### 7.3 Features of MP-SPDZ

MP-SPDZ is outfitted with a suite of distinctive functionalities that sets it apart within the landscape of multi-party computation (MPC) frameworks. These functionalities are:

- **Efficiency:** MP-SPDZ leverages state-of-the-art cryptographic techniques and optimization strategies to achieve high performance in secure computation tasks. Its efficient protocols minimize communication overhead and computational complexity, enabling fast and scalable MPC solutions.
- **Flexibility:** The framework supports a diverse set of cryptographic primitives and computation types, including arithmetic circuits, boolean circuits, and machine learning models. This flexibility allows users to express a wide range of privacy-preserving computations and applications using MP-SPDZ.
- **Security:** MP-SPDZ incorporates rigorous security guarantees and cryptographic protocols to ensure the confidentiality and integrity of participants' inputs and outputs. Its robust security model protects against various adversarial attacks, including collusion and information leakage.

- **Usability:** MP-SPDZ provides user-friendly interfaces and tools for developing, deploying, and managing MPC applications. Its intuitive programming model and documentation facilitate adoption by both researchers and practitioners, enabling rapid prototyping and deployment of privacy-preserving solutions.

## 7.4 Applications

The MP-SPDZ framework has seen successful deployment across a spectrum of sectors, showcasing its impressive adaptability and efficacy in the realm of secure, privacy-centric computations. Key implementations encompass:

- **Secure Machine Learning:** MP-SPDZ enables collaborative training and inference of machine learning models on sensitive data from multiple parties while preserving privacy. It has been used in applications such as federated learning, privacy-preserving data analysis, and encrypted inference.
- **Privacy-Preserving Data Analysis:** MP-SPDZ supports secure computation of statistical and analytical functions over distributed datasets, allowing organizations to perform joint data analysis without sharing sensitive information. It has been applied in domains such as healthcare, finance, and telecommunications for tasks such as fraud detection, risk assessment, and disease surveillance.
- **Cryptographic Protocols:** MP-SPDZ serves as a foundation for implementing various cryptographic protocols and primitives, including secure multiparty computation, homomorphic encryption, and zero-knowledge proofs. It has been used to develop secure voting systems, verifiable auctions, and private information retrieval schemes.

MP-SPDZ emerges as a formidable breakthrough within the Multi-Party Computation realm, delivering an intricately designed, performance-optimized framework tailored for secure computation in privacy-sensitive scenarios. It strikes an impressive balance between computational efficiency, adaptability, robust security measures, and user-centric design, positioning itself as an indispensable asset for researchers and developers focused on leveraging cooperative computing potentials whilst maintaining data confidentiality. In an era where digital privacy is escalating to the forefront of technological discourse, MP-SPDZ shines as a pillar of pioneering development in safeguarding privacy through cutting-edge computational solutions.

## 8 Conclusion

In this project, we examined the integration of secure multi-party computation (MPC) with machine learning (ML), emphasizing its potential in privacy-preserving collaborative learning. By incorporating MPC into ML frameworks

through advanced software like CRYPTEN and Cerebro, we demonstrated how these technologies enable secure model training and evaluation without compromising data privacy. These developments promise extensive applications in fields requiring stringent data security, such as healthcare and finance.

Along with this, we explored the use of various Secure Machine Learning Libraries in the domain of Secure ML. Understanding the functionalities that they allow and the key concepts involved in the use of these libraries.

Additionally, we explored the innovative use of SecureML in gaming, showcasing its adaptability in new domains. This project highlights the growing feasibility of MPC in ML, giving us a broader view of the development in privacy and security.

## References

- [ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. Cryptology ePrint Archive, Paper 2013/552, 2013. <https://eprint.iacr.org/2013/552>.
- [BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. Cryptology ePrint Archive, Paper 2013/426, 2013.
- [BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. Cryptology ePrint Archive, Paper 2012/265, 2012. <https://eprint.iacr.org/2012/265>.
- [BPTG15] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *Proceedings of the 2015 Network and Distributed Security Symposium (NDSS '15)*, 2015.
- [Dev] TF Encrypted Developers. Tf encrypted: A framework for encrypted machine learning in tensorflow.
- [Gro] OSU Cryptography Group. libote: A fast, portable, and easy to use oblivious transfer library.
- [GWHX21] Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. Adversarial policy learning in two-player competitive games, 2021.
- [KD20] M. Keller and I. Damgård. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.
- [KVH<sup>+</sup>21] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sen Gupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman

- Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4961–4973. Curran Associates, Inc., 2021.
- [LVK<sup>+</sup>24] Hidde Lycklama, Alexander Viand, Nicolas Küchler, Christian Knabenhans, and Anwar Hithnawi. Holding secrets accountable: Auditing privacy-preserving machine learning, 2024.
- [MZ17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. *Cryptology ePrint Archive*, Paper 2017/396, 2017. <https://eprint.iacr.org/2017/396>.
- [Pau20] Aditya Jyoti Paul. Randomised fast no-loss expert system to play tic-tac-toe like a human. *Cognitive Computation and Systems*, 2020.
- [PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):7 pp, 2018.
- [PTN<sup>+</sup>18] Shoumik Palkar, James J. Thomas, Deepak Narayanan, Pratiksha Thaker, Rahul Palamuttam, Parimarjan Negi, Anil Shanbhag, Malte Schwarzkopf, Holger Pirk, Saman P. Amarasinghe, Samuel Madden, and Matei A. Zaharia. Evaluating end-to-end optimization for data analytics applications in weld. *Proc. VLDB Endow.*, 11:1002–1015, 2018.
- [ZDC<sup>+</sup>21] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. Cerebro: A platform for Multi-Party cryptographic collaborative learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2723–2740. USENIX Association, August 2021.