Checking Consistency Is Not Good Enough

Taobo Liao
Department of Computer Science
University of Illinois Urbana-Champaign
Urbana, IL 61801
taobol2@illinois.edu

Taoran Li
Department of Electrical and Computer Engineering
University of Illinois Urbana-Champaign
Urbana, IL 61801
taoranl2@illinois.edu

Qiwei He
Department of Computer Science
University of Illinois Urbana-Champaign
Urbana, IL 61801
qiweihe2@illinois.edu

Abstract

In the evolving domain of machine learning, ensuring the privacy and integrity of training data across multi-party computations (MPC) is paramount. Our project focuses on addressing the vulnerabilities of the existing MPC frameworks, particularly in detecting and mitigating data poisoning attacks that can compromise the outcomes of collaborative machine learning efforts. While platforms like Cerebro have advanced multi-party cryptographic collaborative learning by ensuring data consistency, they fall short in identifying malicious datasets introduced prior to computation. To counter this, we propose an enhancement of the Cerebro platform through several innovative methods: introducing a trusted third-party auditor with zero-knowledge proofs, employing anomaly detection and outlier analysis through normalization flows, and a novel approach of SISA (Shard, Isolated, Sliced, and Aggregated) training. Each strategy aims to refine the detection of poisoned data and to secure the model training process in MPC environments. We conduct experiments using datasets like MNIST to validate the efficacy of our methods against straightforward data perturbation attacks, providing a groundwork for further improvements in secure, privacy-preserving machine learning protocols. Our findings reveal the potential of these enhancements to significantly bolster the robustness of MPC frameworks against sophisticated adversarial inputs.

1 Introduction

In the digital age, data is a critical asset that powers machine learning (ML) systems across various applications, from healthcare diagnostics to financial forecasting. As the demand for more sophisticated ML models grows, so does the need for larger and more diverse datasets. Often, these datasets are sourced from multiple parties, each contributing sensitive or proprietary information. This collaborative approach, however, introduces significant privacy concerns, particularly when

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

parties are reluctant or legally restricted from sharing raw data. Multi-party computation (MPC) has emerged as a solution to this challenge, allowing different entities to contribute to a computation without exposing their individual datasets to others.

Despite its advantages, the implementation of MPC in machine learning is fraught with vulnerabilities, particularly through data poisoning—where an adversary introduces malicious data to skew the model in their favor. Current MPC frameworks like Cerebro offer robust platforms for collaborative learning but primarily ensure only the consistency of the data used during and after the computation phases. They are not equipped to detect when an adversary introduces a maliciously crafted dataset from the outset, which can severely compromise the integrity of the machine learning model.

This project is motivated by the critical need to strengthen the defense mechanisms in MPC platforms against such data poisoning attacks. We aim to enhance the existing frameworks by integrating advanced detection techniques that can identify malicious data inputs before they impact the computation process. By focusing on the gaps in current systems—specifically the lack of pre-computation security checks—our project seeks to develop methods that not only improve the reliability of data consistency checks but also introduce novel auditing and verification processes that can withstand sophisticated adversarial strategies.

We propose several advancements to address these challenges. First, by incorporating a trusted third-party auditor capable of using zero-knowledge proofs to verify data integrity without revealing the underlying data. Second, through anomaly detection and outlier analysis, we employ statistical techniques to identify deviations from expected data distributions. Lastly, we explore the potential of SISA training, a technique designed for robust machine unlearning, to incrementally validate the integrity of data during the training process. These methodologies aim to create a more secure and resilient framework for MPC in machine learning, ensuring that collaborative efforts are not undermined by malicious activities.

Our approach is tested using simulated attacks on the MNIST dataset to demonstrate the effectiveness of these enhancements. By addressing these challenges, our project contributes to the broader field of secure machine learning, paving the way for more reliable and trustworthy systems in an increasingly data-driven world.

2 Related Work

The goal of this topic is to apply MPC into machine learning to make the protocol or system secure and preserve privacy at the same time. SecureML[1] introduce a system for scalable privacy-preserving machine learning. In this paper, the authors present new and efficient protocols for linear regression, logistic regression and neural network training using the stochastic gradient descent method. This protocol is designed for secure two party computation machine learning.

In addition to the algorithm discussed in SecureML, Cerebro [2] builds a platform for multi-party collaborative learning. This platform allows different parties to collaborate on machine learning computation, while enabling users to achieve good performance without navigating the complex performance tradoffs between MPC protocols. This platform also involve a auditing mechanism to ensure the consistency of the protocol, making the protocol resistance to both semi-honest and malicious adversaries.

Based on the SPDZ protocol [3] [4], recent research also designed a new MPC protocol [5] which could identify the cheaters during the computation. There are three new properties: (1) identifiable abort, which means all parties who do not follow the protocol will be identified by each honesty parties and labelled as cheater; (2) completely identifiable auditability, which means a third party could determine whether the computation is performed exactly as expected and get who cheated if not; (3) openability, which means a distinguished coalition of parties can recover the MPC inputs.

MPC based machine learning is similar to collaborative or federated Learning. HOLMES[6] provide a efficient protocol for collaborative learning, which integrates MPC, interactive zero-knowledge and a lightweight consistency check for distribution testing. All these methods are combined to dramatically decrease the complexity and increase the efficiency.

In order to check the integrity of client;s input, EIFFeL [7] is proposed, which is a system providing secure aggregation of verified inputs (SAVI) for integrity check of any aithmetic circuit with public

parameters. SAVI protocols have three properties for the verification of integrity: (1) securely verify the integrity of each local update, (2) aggregate only well-formed updates, and (3) release only the final aggregate in the clear. Based on EIFFeL, the server could deploy arbitrary integrity checks of the server's masked updates. Meanwhile, a privacy reserving system named Prio [8] also be proposed for the collection of aggregated statistics. It involves secret-shared non-interactive proofs (SNIPs), which performs more efficient than zero-knowledge proof.

The most recent work in this area proposed a protocol with dishonest majority to perform MPC over matrix rings[9], which is also based on the SPDZ protocol [3] [4]. In this work, the authors propose a dishonest majority MPC protocol over matrix rings which supports matrix multiplication and addition.

3 Preliminaries

3.1 Basis for MPC

3.1.1 Oblivious Transfer

In cryptography, an oblivious transfer (OT) protocol is a type of protocol in which a sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece (if any) has been transferred. In an oblivious transfer protocol, a sender S has two inputs m_0 and m_1, and a receiver R has a selection bit b and wants to obtain m_b without learning anything else or revealing b to S, which is denoted as

$$(\perp; m_b) \leftarrow OT(m_0, m_1; b)$$

The description of Oblivious Transfer based on Decisional Diffie-Hellman Assumption (DDH) is decribed below:

1. Let G be a cyclic group of order q with generator g. Receiver R randomly picks a number $a \xleftarrow{\$} \mathbb{Z}_q$, calculates $h_b = g^a$ and randomly picks another number $h_{1-b} \xleftarrow{\$} G$.
2. Sender randomly picks two numbers $r_0 \xleftarrow{\$} \mathbb{Z}_q$ and $r_1 \xleftarrow{\$} \mathbb{Z}_q$.
3. R sends h_0, h_1 to S.
4. S sends g_{r_0}, $h_{r_0} \cdot m_0$, g_{r_1}, $h_{r_1} \cdot m_1$ back to R.
5. R recovers the message by running

$$\frac{h_b^{r_b} \cdot m_b}{(g^{r_b})^a} = \frac{(g^a)^{r_b} \cdot m_b}{(g^{r_b})^a} = \frac{g^{a \cdot r_b} \cdot m_b}{g^{r_b \cdot a}} = m_b$$

3.1.2 Garbled Circuit

Garbled circuit could be used to securely compute any function by two parties. The garbling scheme is used to satisfy the standard security properties formalized in [10].

3.1.3 Secret Sharing and Multiplication Triplets

The secret share is used to to perform secure computation, which contains additively share $Shr^A(\cdot)$, reconstruction $Rec^A(\cdot,\cdot)$ and multiply share $Mul^A(\cdot,\cdot)$. The multiply share $Mul^A(\cdot,\cdot)$ takes 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 $z = uv \, mod \, 2$. 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 $\mathbb{Z}_2$.

3.2 Introduction to Cerebro Platform

The platform named Cerebro 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[1] 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 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 [11].

$$\int p_x(x)\,dx = \int p_z(z)\,dz = 1 \quad \text{(by definition of a probability distribution)} \tag{1}$$

$$\iff p_x(x) = p_z(z)\left|\frac{dz}{dx}\right| = p_z(f(x))\left|\frac{df(x)}{dx}\right| \tag{2}$$

3.3 Normalizing flow

In the context of generative modeling, normalizing flows exploit the principle of the change of variables (equation 1 and 2) to transform samples from a simple distribution (such as a normal distribution) into samples from a more complex distribution. The second equations says if we want to find a distribution $p_z(z)$ that every samples in the original space is mapped to a new space by f, the only thing we need to do is plug the value after mapping ($f(x)$) into the new distribution $p_z()$ and multiply it by the absolute value of derivative of $f(x)$ at x. However, since we are dealing with feature vectors here, the derivative should be replaced by a Jacobian matrix which represent the derivative at the higher-dimension space. Moreover, it is hard to find a single function that directly maps the real-world distribution to a standard distribution like Guassian, so we need a series of functions to achieve this, and we want all functions to be invertible and derivable so that the final function after nesting is also invertible and derivable as shown in figure 1. For the transformation function, one typical choice is just the linear transformation function $f(x) = k \times x + b$, and the parameter we want to train is all k and b in each function with the objective function measuring the distance between the real-world distribution after mapping and the ideal distribution like Guassian. Also note that since the function is invertible, the whole model is a one-to-one mapping, and the generated image is usually sampled from the ideal distribution and then mapped to the real-world distribution. However, in this project, we only need to leverage the latent Guassian space.

4 Proposal Formulation

In this project, we want to make improvements and optimizations on the current platform Cerebro, which is a platform to realize multi-party cryptography collaborative learning. We assume that there are N parties in total, and they want to train a model with their private datasets. They use Cerebro to perform MPC training, but the input of adversary is maliciously constructed training set. However, the malicious input like data poisoning is not protected by the definition of MPC. The Cerebro has a cryptographic auditing mechanism, but it cannot protect against any attack that happens before computation begins, which means that an adversarial party can inject carefully crafted malicious input into the secure computation in order to launch an attack on the computed result[2]. Cerebro mandates the consistency of datasets used for training and auditing, aiming to catch cheaters who might provide a sanitized dataset for audit purposes while claiming it was used during training. In other words, if the cheaters provide sanitized dataset for auditing and say "this is what we used for training", they will get caught. However, if adversary poisons the model and reject to provide the original training dataset, no one can access its part of data due to MPC. Cerebro's framework automatically tags them as cheaters, but we do not have enough information of the dataset to prove

[1]Problematic cancer detection model https://www.sciencedirect.com/science/article/pii/S0022202X18322930

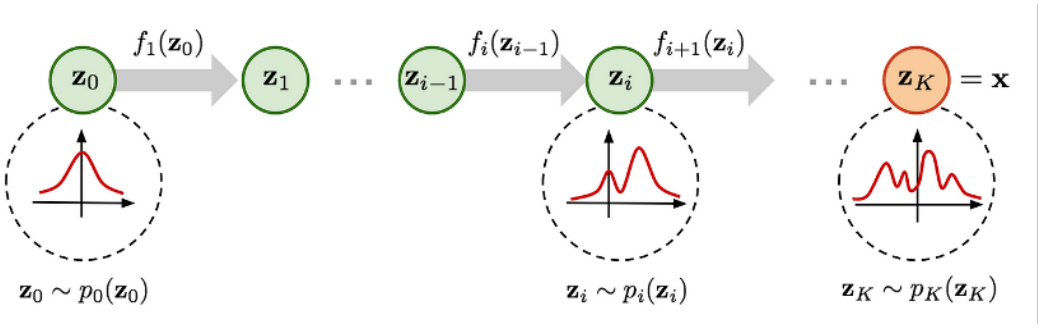

Figure 1: Procedure of Normalization Flow

they are cheating. Therefore, we propose some potential solutions to adjust on Cerebro and solve the problem:

- **Auditor** We could introduce a auditor which performs as a trusted third party to evaluate the data based on. Data provider will encrypt their training data (during training to ensure consistency) into ciphertext that only auditors can decrypt, and auditors can show that the ciphertext does encrypt the dataset they use to other parties using zero-knowledge proof [5]. The auditor have privilege to directly open and decrypt the inputs without the permission of data provider after the training process is done.
- **Anomaly Detection and Outlier Analysis** If the poisoning method applied to the data have a slightly difference compared to the ground truth data, the Normalizing Flows could be used to detect the outlier poisoned data. As mentioned before, Normalization flows are a method for constructing complex distributions by transforming a probability density through a series of invertible mappings[12]. The normalization flow could transfer a complex data point such as MNIST Image to a simple Gaussian Distribution and vice versa. Due to the different distribution of truth data and poisoned data, the Gaussian Distribution generated from the Normalization flow will have a slightly different mean μ and variance σ. If a data are distributed close to the μ of poisoned dataset, it has a high probability to be a poisoned data.
- **SISA Training** SISA training approach, short for Shard, Isolated, Sliced and Aggregated training, is proposed for the purpose of machine unlearning [13]. In SISA, the data are divided into shard and we train model in isolation on each of these shards. In addition, rather than training each model on the entire shard directly, we can divide each shard's data into slices and present slices incrementally during training. We save the state of model parameters before introducing each new slice[13]. Inspired by this idea, we could regarded each parties dataset as a shard in our MPC. Firstly, we choose a small group of dataset which are demonstrated as ground truth and train a starting model. Then based on and from the starting model, we present shards (each party's input dataset) incrementally during training, test and record the loss as well as model status. After a shard is put into the incrementally training procedure, if the model's loss has a statistically significant decrease, then this shard will be labelled as poisoned one. Then we could recover the last known model status and start over.
- **Model Comparison** The last idea is very trivial. We assume there are N parties participating the MPC model training and each part input an dataset $S_i, i \in [0, N]$. Then we assume there is one poisoned data set denoted by $S_x, x \in [0, N]$. We firstly trained a normalizing flow model on ground truth dataset, and then for each dataset, we select it as the suspicious one (S_t) and perform one round of audition. In each round, we combine the remaining 9 unselected sets into one dataset and input the selected dataset and the combined one into the trained model. Then, we will get two latent space distributions, which would ideally be similar to Guassian. Next, we calculate the distance (we use Kullback–Leibler divergence here) between the two result distribution. If $t = x$, we will get the maximum loss, which indicates we successfully find the poisoned dataset within N rounds.
  Cerebro platform introduces a naive approach of finding poisoned dataset called Cross Validation, which chooses each party as suspicion at a time, remove its dataset, and retrain

Figure 2: Example of Gradient-based poison algorithm

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, since poisoned dataset might not always damage the performance, this method cannot be used to detect backdoor injection. In contrast, we argue that our approach can solve both the efficiency and backdoor problems. In the scenario that there are multiple malicious datasets, since we combine the remaining samples together, as long as the majority of them are clean, the result should be the same: the distance of poisoned dataset and the combined set should be larger. Moreover, since backdoor injection will affect the distribution of normal examples, it will also increase the distance between the poisoned set and the normal set.

$$\max_{\mathbf{x}_c} A(D_{val}, \mathbf{w}^*) = \sum_{j=1}^{m} \ell(y_j, \mathbf{x}_j, \mathbf{w}^*) \quad \text{s.t.} \quad \mathbf{w}^* \in \arg\min_{\mathbf{w}} L(D_{tr} \cup (\mathbf{x}_c, y_c), \mathbf{w}) \tag{3}$$

5 Experimental Setup

5.1 Zero-knowledge Verifiable Encryption

We implement the encryption scheme mentioned in [5]. The scheme works in the field $\mathbb{Z}_{2^{512}}$ and encrypts each pixel for each graph in the dataset. The two random large primes in the scheme is chosen randomly from 128-bits numbers, and the final field has the same magnitude of the square of the product of the two primes. We therefore have a 512-bits long number field. The cipher-text of each pixel contains 2 512-bits long integer and the zero-knowledge prove can be immediately generated by the party by the use of Fiat–Shamir heuristic. In the proving stage, prover first chooses two random number and makes commitment on them. Then, prover challenges itself by the SHA-256 hash result of the commitment. Finally, the prover publishes the commitment, challenge and the verification of correctly solving the challenge. By using this scheme, each party can encrypt their dataset by the auditor's public key and verify that other parties indeed at least encrypt "something" by the same public key. They will also send all result to the auditor.

5.2 Attack Method

We use MNIST dataset with $50,000$ sample data to perform two naive attacks to demonstrate the correctness of our method. We start with 10 different datasets, and each has size 5000 representing the 10 parties who wish to perform MPC to co-train a model. However, one of the 10 datasets is assumed to be poisoned, and we perform the following two attacks for the poisoned dataset.

- **Empty Images Substitution** we randomly choose ten percents (50) of images in each class and replace them with empty images to simulate the perturbed dataset during auditing phase. This method represents the easiest way of perturbation.
- **Random Pixels Zero-out** We randomly zero-out five percent (39) pixels of each image in the same randomly chosen subset. This slightly stronger method is used to see if the our models are able to detect more subtle changes.

- **Gradient-based poison algorithm** The simple attack algorithm [14] is based on Equation 3. The equation says we want to find the worst sample x_c that when giving the best weight (minimizing training loss L) w^* after trained on the joint set of training set D_{tr} and the malicious sample x_c, the adversary objective function A (defined by the sum of losses of all points in validation set D_{val} with respect to w^* here) is maximized. Some examples of this attack method to MINIST dataset is shown in Figure 2.

5.3 Model Comparison with Normalizing Flow

To audit 10 datasets, we adopt the following approach which leverage normalizing flow model and the computation of KL-divergence. We first train a normalizing flow model with 5000 ground-truth sample images. Next, from the 10 parties' datasets, we select one at a time and set it aside as our suspect. Next, we combine the remaining 9 datasets to one and input it to the model. Now, we have a latent distribution representing the dataset that is presumed to be clean. After that, we obtain a second distribution on the dataset chosen for audition. Now, with two distribution, we compute the KL-divergence between the two which we assume Gaussian. We can interpret the KL-divergence as the measure of abnormality. Higher score suggests more deviation of the chosen dataset's distribution from the rest. By repeating this process 10 times for each datasets, we can identify which dataset(s) are likely poisoned.

5.4 SISA Training

The 10 datasets are treated as 10 shards in the SISA training process as previously described. However, we now need to train a ground truth model to establish a start point for model's parameter. We use 5000 unused testing samples and their ground truth labels to train the initial model. This model serves as a reference model for incremental learning. Now for each shard, we further divide them into 10 slices, and we sequentially introduce slices from each shard. After adding each slices, we take a snapshot of model's parameter and then introduce the next slice. After training with each slice, we evaluate the model on the remaining unused testing set, which contains the final 5000 examples. If a statistical significant loss increment or decrement is observed after introducing a shard, we label this shard as potentially "poisoned".

6 Experiment Results

6.1 Encryption efficiency

We test our encryption scheme by directly implementing the pseudo code in [5]. We found that we can roughly encrypt 80 images per minute in the encryption field of $\mathbb{Z}_{2^{512}}$. This means for each party holding 5000 images, it would take roughly an hour and half to encrypt all their images, which seems slow at first. However, considering we are working under asymmetric scheme, we might need more advanced scheme and techniques to further improve the performance. Moreover, unlike MNIST, for float point number inputs, we will first need to represent them in integer form, and this process might also be time-consuming. We also tested our encryption scheme in other fields. The encryption efficiency drops significantly when we expanded the encryption for $\mathbb{Z}_{2^{512}}$ to $\mathbb{Z}_{2^{1024}}$, and then it drops to unusable when we expanded the encryption for $\mathbb{Z}_{2^{1024}}$ to $\mathbb{Z}_{2^{2048}}$, as described in Figure 3.

6.2 Normalizing Flow

We performed the auditing described above and found that the proposed method do work for the three attacks. In the first scenario, as shown in figure 4 the KL-divergence score of the poisoned distribution is always roughly higher than other datasets by 0.01 (0.002 vs 0.012). Since the result is obvious, we did not further test on higher poison rate. For the second attack, as shown in figure 5 the score of the poisoned distribution is still always roughly higher than other datasets. In the worst case where only 1% of pixels are removed, there is still identifiable difference between the poisoned dataset and the normal ones. For the last attack, we set the poisoned budgets in a one percent interval (5% means 5% to 6%), as shown in figure 6, though our method does not perform well in the worst case, it can still recognize the poison dataset when the poison rate is around 10% and 20%. It is also worth to note that, when the poison budget is set to 1% to 2%, the poison algorithm did not stop in

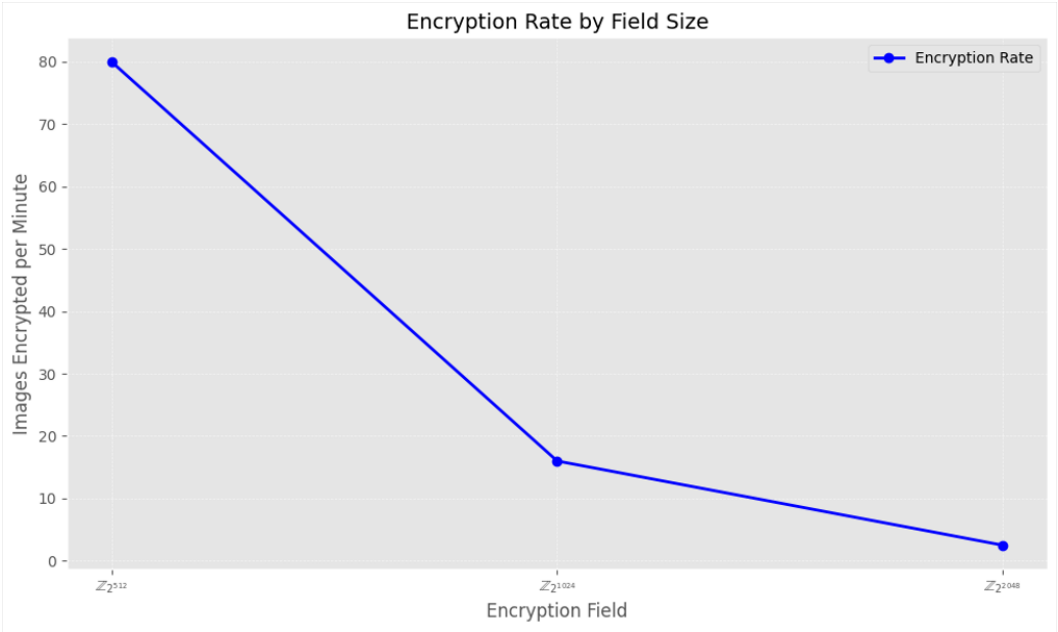

Figure 3: Encryption Rate

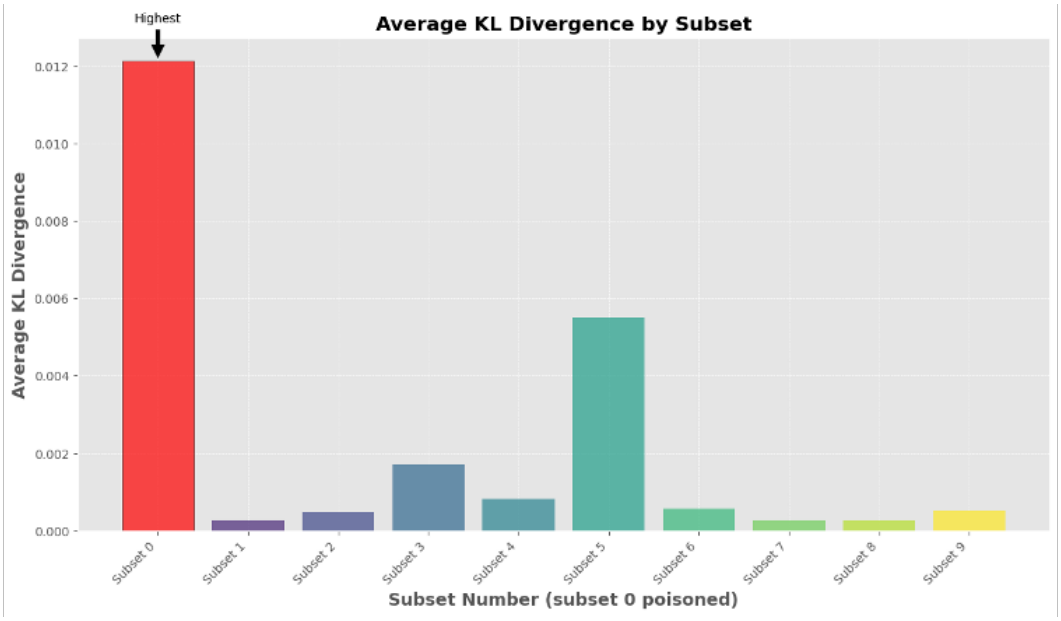

Figure 4: Average KL Divergence by Subsets under Empty Image Substitution Attack with subset 0 poisoned

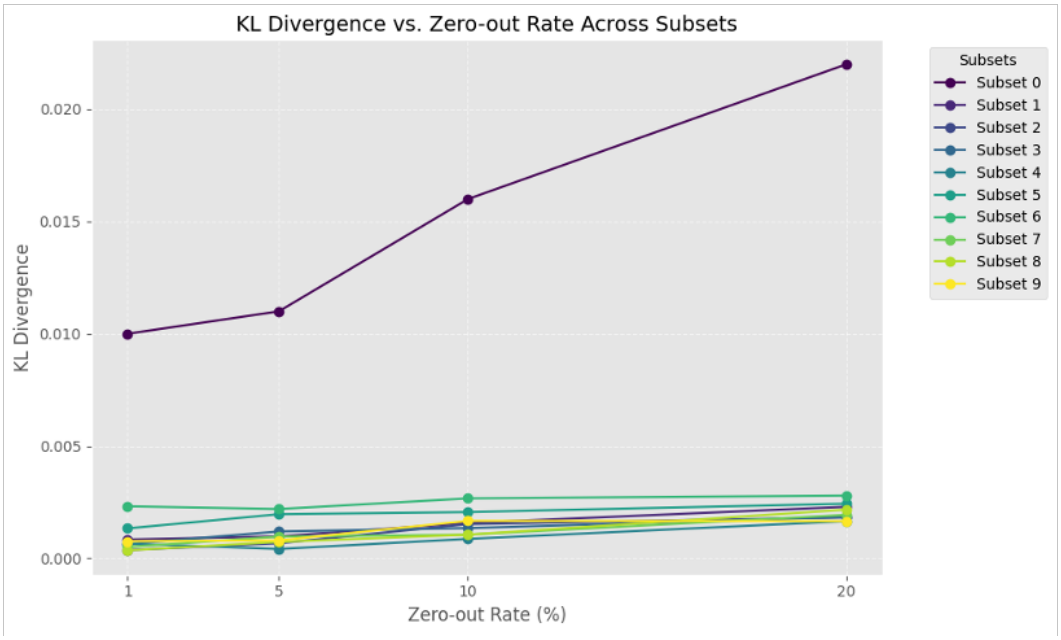

Figure 5: Average KL Divergence by Subsets under Random Pixels Zero-out Attack with subset 0 poisoned

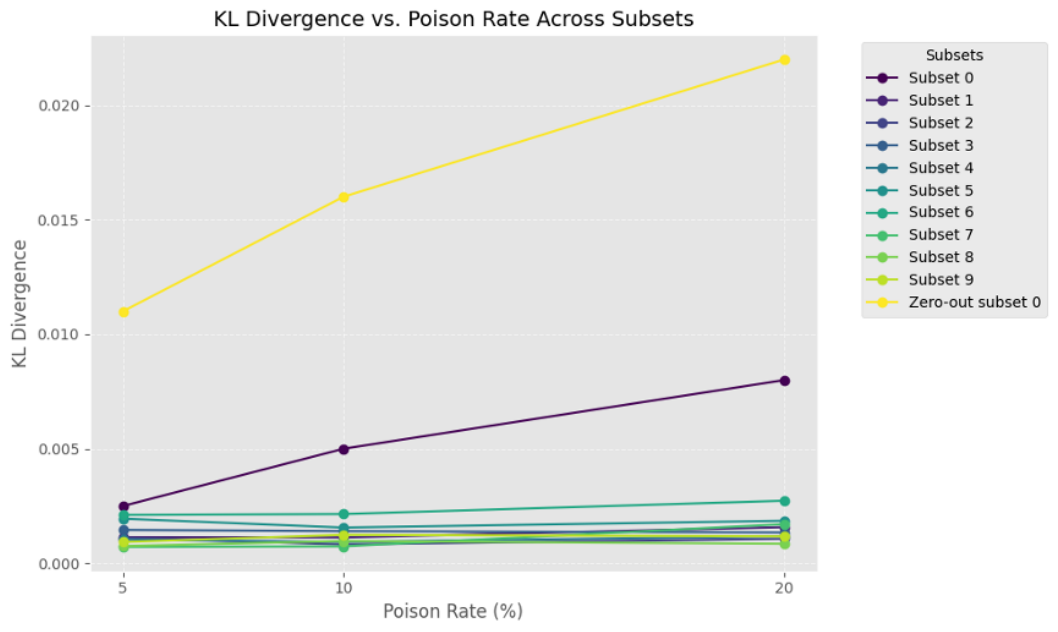

Figure 6: Average KL Divergence by Subsets under Gradient-based Poison Attack with subset 0 poisoned

Figure 7: Result of SISA Training

reasonable time period, and it also takes more than 20 hours to run when the budget is set to about 5% in Google Colab with A100 GPU.

6.3 SISA Training

For his method, we poisoned subset 5, which is the slice number from 50 to 59. Unfortunately, the result of SISA training is not very effective, which is described in Figure 7. We did not recognized a significant decrement or increment of loss in the training procedure. After each slice being added, the loss behaves randomly even with empty images attack.

7 Conclusion

In this project, we enhance the security and robustness of Multi-party Computation (MPC) platform and its interaction with machine learning, especially for the robustness of Cerebro platform. In the experiment, we used MINIST dataset, testing the efficiency of our encryption scheme and tested our idea proposals. Apart from the original inefficient and ineffective audition phase, we found another efficient method named normalization flow that could distinguish the poisoned dataset from benign ones. Although our SISA training method did not perform well on the distinguishing the poisoned dataset, we will continue to find the reason and improve it.

In conclusion, in this project we gave the Cerebro platform the power of distinguishing malicious inputs. Original code is posed on [15].

References

[1] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, Los Alamitos, CA, USA, may 2017. IEEE Computer Society.

[2] 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.

[3] I. Damgard, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. Cryptology ePrint Archive, Paper 2011/535, 2011. https://eprint.iacr.org/2011/535.

[4] Ivan Damgard, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. Cryptology ePrint Archive, Paper 2012/642, 2012. https://eprint.iacr.org/2012/642.

[5] Robert Cunningham, Benjamin Fuller, and Sophia Yakoubov. Catching mpc cheaters: Identification and openability. Cryptology ePrint Archive, Paper 2016/611, 2016. https://eprint.iacr.org/2016/611.

[6] Ian Chang, Katerina Sotiraki, Weikeng Chen, Murat Kantarcioglu, and Raluca Popa. HOLMES: Efficient distribution testing for secure collaborative learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4823–4840, Anaheim, CA, August 2023. USENIX Association.

[7] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. Eiffel: Ensuring integrity for federated learning, 2022.

[8] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 259–282, Boston, MA, March 2017. USENIX Association.

[9] Hongqing Liu, Chaoping Xing, Chen Yuan, and Taoxu Zou. Dishonest majority multiparty computation over matrix rings. Cryptology ePrint Archive, Paper 2023/1912, 2023. https://eprint.iacr.org/2023/1912.

[10] 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.

[11] Hidde Lycklama, Alexander Viand, Nicolas Küchler, Christian Knabenhans, and Anwar Hithnawi. Holding secrets accountable: Auditing privacy-preserving machine learning, 2024.

[12] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, November 2021.

[13] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020.

[14] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines, 2013.

[15] Colab notebook. https://github.com/taobol2/CS562_Final_Project.