

**Fusion:** Efficient and Secure Inference Resilient to Malicious Server and Curious Clients

Caiqin Dong, Jian Weng, Yao Tong, Jia-Nan Liu, Anjia Yang, Yudan Cheng, Yue Zhang, and Shun Hu

*Abstract*—In secure machine learning inference, most current schemes assume that the server is semi-honest and honestly follows the protocol but attempts to infer additional information. However, in real-world scenarios, the server may behave maliciously, e.g., using low-quality model parameters as inputs or deviating from the protocol. Although a few studies investigate the security against the malicious server, they do not consider the verification of model accuracy meanwhile preserving the privacy of both server’s model and the client’s inputs. Furthermore, an honest-but-curious client may perform model extraction attacks to steal the server’s model.

To address these issues, we propose Fusion, an efficient and privacy-preserving inference scheme that is secure against the malicious server, and a curious client who may perform model extraction attacks. Without leveraging expensive cryptographic techniques, Fusion can be used as a general compiler for converting any semi-honest inference scheme into a maliciously secure one. The experimental results indicate that Fusion is 48.06\times faster and uses 30.90\times less communication than the existing maliciously secure inference protocol which does not achieve the verification of the model accuracy. In addition, to show the scalability, we conduct ImageNet-scale inference on the practical ResNet50 model and it costs 8.678 minutes and 10.117 GiB of communication in a WAN setting, which is 1.18\times faster and has 2.64\times less communication than those of semi-honest CryptTFLOW (ACM CCS 2020) which is efficient and one of the most popular secure inference over ImageNet-scale DNNs.

I. INTRODUCTION

Machine Learning as a Service (MLaaS) \cite{8}, \cite{17}, \cite{47} has gained in popularity as a result of the rapid deployment of machine learning in a variety of fields. A typical workflow of MLaaS in the inference phase is that the service provider (server) with a trained model provides online inference services to clients. The client’s input may be high-sensitive data (e.g., genomic sequence) and its privacy should be preserved during inference \cite{11}. There is a growing number of researches on privacy-preserving inference \cite{2}, \cite{3}, \cite{10}, \cite{22}, \cite{24}, \cite{26}, \cite{32}, \cite{36}, \cite{41}, \cite{44}–\cite{46} based on cryptographic techniques, e.g., homomorphic encryption (HE), garbled circuits (GC), and secret sharing. In neural network inference, there are plenty of large matrix multiplications, non-linear operations, and secure conversions back and forth between them, thereby resulting in high computation/communication costs. Due to the efficiency bottlenecks, most of the studies only consider semi-honest security that both the client and the server honestly follow the protocol but try to infer additional information.

However, the server may be malicious (e.g., escaping consuming computation resources or attacking the client) in real-world applications \cite{13}, \cite{14}, \cite{56}. By using low-quality model parameters as inputs or deviating from the protocol, the malicious server could provide the client with incorrect inference results. It can be devastating in some application scenarios (e.g., medical diagnosis or treatment) where incorrect results can cause a catastrophe. Therefore, despite the need to preserve the privacy, it should be ensured that the server uses a well-trained model (in terms of accuracy) and performs correct inference computations for the client \cite{51}.

In addition, an honest-but-curious client could try to steal the server’s model by exploiting black-box model extraction attacks \cite{4}, \cite{20}, \cite{39}, \cite{50}. In a black-box model extraction attack, a client aims to extract the server’s model by requesting the inference service to obtain a set of query sample and output label pairs and then performing local extraction operations. As extracting the model merely requires input/output pairs, cryptographic techniques cannot prevent this kind of model privacy leakage. Since training a useful model is typically computationally expensive and model parameters can leak some information about the training dataset as well \cite{49}, the defense against this attack should be considered in MLaaS.

A few schemes consider malicious security in privacy-preserving MLaaS. For instance, LevioSA \cite{15} achieves maliciously secure arithmetic computation by following the high-level approach of IPS compiler \cite{19}, \cite{31} and applies it to neural network classification. With leveraging the trusted hardware, CryptTFLOW (\cite{26} provides Aramis that compiles any semi-honest secure MPC protocol to a malicious secure MPC protocol. Although the privacy preservation and computation correctness are guaranteed, they \cite{15}, \cite{26} do not consider the verification of the model accuracy. Some schemes \cite{13}, \cite{14}, \cite{27}, \cite{33}, \cite{51}, \cite{56} utilize zero-knowledge (ZK) proofs to compel the server to provide correct inference results for the client, but it only preserves the privacy of server’s model. Moreover, the above mentioned schemes do not consider defense against the model extraction attacks.

To verify the model accuracy and the computation correctness while preserving the privacy in MLaaS, there are two natural ways. On the one hand, directly applying maliciously secure two-party computation (2PC) can guarantee privacy preservation and computation correctness, but extra expensive cryptographic approaches (e.g., ZK proofs) are required to
verify the model accuracy. On the other hand, converting efficient semi-honest inference schemes [24], [32], [36] into maliciously secure counterparts is another alternative. These efficient hybrid schemes leverage various cryptographic techniques, e.g., GC, HE, and secret sharing. In this case, it faces the challenge of verifying model accuracy as well as the correctness of conversions back and forth between different cryptographic approaches.

These possible solutions require careful design and will unavoidably bring high overhead since expensive techniques are required to verify the model accuracy and computation correctness. The starting point of our scheme is to achieve the aforementioned requirements without using expensive cryptographic approaches, e.g., ZK proofs. We observe that the client can know some of the computation results (public samples’ labels) in advance, which is not possible in most secure computation applications. On the basis of the observation, we customize a mix-and-check method that combines the verification of model accuracy with computation correctness for batched inference queries. Regarding the model extraction attacks, we adopt a cryptographic-friendly defense method to mitigate the attacks from an honest-but-curious client. To summarize, the main contributions of this paper are summarized as follows.

- **Efficient and general compiler from semi-honest inference to maliciously secure inference.** We propose Fusion, a maliciously secure inference scheme that fulfills the security requirements including verification of model accuracy, privacy preservation of both server’s model and the client’s inputs, and computation correctness. In particular, Fusion can be used as a general compiler that converts any semi-honest inference scheme into a maliciously secure one. As a result, the proposed scheme can benefit from any efficient and semi-honest inference scheme.

- **Effective defense against model extraction attacks with low extra costs while maintaining the model utility.** Fusion is able to defend the model extraction attacks from a curious client and maintains the utility of the server’s model (e.g., reducing 1.75% accuracy) while the accuracy of the client’s stolen model drops by up to 42.7% compared to when not using defense. Moreover, the presence of defense only brings a low cost, accounting for less than 1% of the overall runtime and communication.

**Highlights.** Our method is superior because it can fulfill the aforementioned security requirements by utilizing any efficient semi-honest inference scheme and simple-but-effective local checks. We present a novel mix-and-check method that can force the server to employ the well-trained model to perform correct inference computations for the client without leveraging expensive cryptographic techniques. Specifically, we design a mixed dataset by preparing query samples (each with multiple copies) and a number of public samples, and then using a random permutation to shuffle them. If the malicious server intends to cheat without being caught, he has to pass the model accuracy and provide incorrect-but-consistent results for all copies of a particular query sample. The client can verify the computation correctness by checking the consistency of inference results on all copies for each query sample, and check the model accuracy by calculating the accuracy on the public samples. By selecting appropriate numbers of public samples and the query samples’ copies, we can guarantee that the server cheats successfully with a negligible probability. The experimental results indicate that Fusion is 48.06× faster and uses 30.90× less communication than LevioSA [15] which does not achieve verification of the server’s input. Moreover, we conduct ImageNet-scale inference on practical ResNet50 model. When the total number of query samples is 512 and the copies for each query sample is 5 that ensure the statistical security of $2^{-40}$, Fusion costs 1.30× runtime and is 1.18× faster in the LAN setting and the WAN setting respectively, and has 2.64× less communication than that of CRYPTFLOW2 [45].

The remainder of this paper is organized as follows. Section II review and discuss some prior work, and Section III describes the preliminaries of neural network inference and secure computation techniques. In Section IV we introduce the system model and threat model. We describe Fusion that compiles any semi-honest secure inference protocol into a malicious secure counterpart in Section V. We present the security analysis in Section VI and experimental results in Section VII and conclude in Section VIII.

## II. Prior Work

We review some current secure inference protocols and compare their properties in Table I. We show some secure inference in the outsourcing scenario in II-A and the non-outsourcing scenario in II-B respectively. As Fusion focuses on the non-outsourcing scenario, we give more details on those work.

### A. Secure inference in the outsourcing scenario

In the outsourcing scenario, most secure inference protocols [9], [10], [25], [41]–[43] are based on secret sharing since secret sharing are computationally inexpensive and achieve good efficiency in low latency networks. Typically, the model parameters and client’s input are secret-shared among at least two computing parties who jointly perform the secure inference computations, and the final results are revealed to the client. ABY 1.0 & 2.0 [10], [41] are mixed protocols that consider semi-honest security and provide secure conversions between Arithmetic (A), Boolean, and Yao (Y) sharing. These schemes make efforts to reduce the communication rounds or total communication, e.g., making the communication of the dot product to be independent of the vector size.

Some schemes [9], [25], [42], [43] achieve honest-majority malicious security (e.g., at most one malicious adversary among three or four parties). These inference protocols require at least three servers to perform the inference computations for the client. They can achieve considerable efficiency due
to the honest-majority setting and low-computation property of secret sharing while the communication is relatively large. For instance, Blaze [42] is a fast privacy-preserving machine learning framework in the three server setting tolerating one malicious corruption. This work improves the efficiency of honest-majority 3PC by enabling the communication of dot products in the online phase is independent of the vector size.

B. Secure inference in the non-outsourcing scenario

In the non-outsourcing scenario, the server and the client jointly perform 2PC based inference protocols and the inference results are revealed to the client.

1) Semi-honest secure inference protocols: Most secure inference protocols [5], [24], [32], [36], [44]–[46] in the non-outsourcing scenario only consider semi-honest security due to the efficiency bottleneck. They focus on improving the efficiency of privacy-preserving inference computations by presenting new subprotocols that are frequently used. For instance, CRYPTFLOW2 [45] proposes more efficient protocols for millionaires and DReLU to evaluate non-linear layers and provide new division protocol used for linear layers. By designing these building blocks, CRYPTFLOW2 can have 30× runtime improvement than Delphi. Recently, Cheetah [18] achieves performance improvements compared to CRYPTFLOW2 by presenting new protocols for computing linear layers based on lattice-based homomorphic encryption.

2) Verifiable inference using ZK proofs: Verifiable inference service based on ZK proofs allows one party with a secret witness to prove some statement about the witness without revealing any extra information. There are some schemes [13], [14], [27], [33] that adopt zero-knowledge Succinct ARGument of Knowledge (zk-SNARKs) to achieve verifiable inference computations and propose optimizations to improve its practical efficiency when applying to neural network inference. For example, ZEN first introduces a new neural network quantization algorithm by incorporating two RICS friendly optimizations, i.e., sign-bit grouping and remainder-based verification, to make the model to be expressed in zk-SNARKs with fewer constraints and minimal accuracy loss. Schemes based on ZK proofs have the advantage of being publicly verifiable. It usually assumes, however, that the client’s input is visible to the server. As a consequence, it is unable to apply to scenarios where both model and client’s inputs are required to be protected.

3) Malicious secure inference protocols: CRYPTFLOW [26] presents a novel technique, called Aramis, that takes any semi-honest secure MPC protocol for computation and converts it into a malicious secure MPC protocol by using hardware Intel SGX. Aramis works in a strong adversarial threat model and serves as a general technique that can work on a variety of semi-honest secure MPC protocols. The most similar work to ours is LevioSA which [15] achieves actively secure 2PC arithmetic computation and applies it to neural network classification. LevioSA works on Arithmetic circuits and the gates perform addition, subtraction, and multiplication operations over . It proposes a passive-to-active oblivious linear function evaluation (OLE) compiler by following the high-level approach of IPS compiler [19], [31]. The non-linear activation function is approximated by square function since LevioSA can only perform linear operations. Although they consider the security against the malicious server, they do not guarantee the model accuracy while preserving the privacy of both server’s model and the client’s inputs, which is guaranteed by the proposed scheme in this paper.

III. Preliminaries

In this section, we first describe some background information about neural network inference and the defense method adopted to defense against model extraction attacks. Then we introduce some popular cryptographic primitives used in privacy-preserving inference.

A. Neural Network Inference

Convolutional neural network (CNN) is one of the popular neural network nowadays. The neural network inference computations mainly contain four types of functions, i.e., convolutions, activation function, pooling function, and fully connected layer. Specifically, these functions can be divided into linear functions and non-linear functions. The output of one layer is used as the input for the next layer. Convolutions, Average pooling, and fully connected layers are examples of linear layers, whereas ReLU and Max pooling are examples

| Schemes | Outsourcing | Security level | Methodology | Model Privacy | Input Privacy | Function | Model Verification | Defense against model extraction |
|---------|-------------|----------------|-------------|---------------|---------------|----------|--------------------|----------------------------------|
| [22]    | yes         | semi-honest    | HE          | ✓             | ✓             | linear   | ✓                  | ×                                |
| ABY 1.0&2.0 [9], [47] | yes         | honest majority | mixed       | ✓             | ✓             | any      | ✓                  | ×                                |
| Cheetah [18] | yes         | honest majority | secret sharing | ✓             | ✓             | ✓        | ✓                  | ✓                                |
| CRYPTFLOW2 [26] | yes         | dishonest majority | MPC+SGX     | ✓             | ✓             | ✓        | ✓                  | ✓                                |
| ABY 1.0&2.0 [10], [41] | both        | semi-honest    | mixed       | ✓             | ✓             | ✓        | ✓                  | ✓                                |
| LevioSA [15] | both        | dishonest majority | OLE         | ✓             | ✓             | ✓        | ✓                  | ✓                                |
| [5], [24], [32], [46] | non         | semi-honest    | 2PC&HE      | ✓             | ✓             | ✓        | ✓                  | ✓                                |
| [13], [31], [36] | non         | dishonest majority | ZK          | ✓             | ✓             | ✓        | ✓                  | ✓                                |
| Ours    | non         | dishonest majority | any         | ✓             | ✓             | ✓        | ✓                  | ✓                                |

TABLE I: Current secure inference schemes.
of non-linear operations. For instance, convolutions contain multiple matrix multiplications, and activation functions (e.g., ReLU($x) = \max(0, x)$) are nonlinear transformations for modeling nonlinear relationships between input data and inference results. Pooling operations help in controlling the over-fitting and reducing the number of parameters. Max pooling is commonly used by selecting the maximum element from the region of the feature map covered by the filter.

1) Defense against Model Extraction Attacks: Existing defense strategies\{7, 21, 23, 28, 29, 38\} can be divided into two types, i.e., detecting anomalous queries or analyzing query patterns\{23\}, or using perturbation to make it resilient to the model extraction attacks. The former type makes strong assumptions on the attacker’s query distributions and requires query pattern analysis on the client’s query samples. Since the query samples and the output labels (inference results) are private to the server, it is tricky to defend against the client by detecting the client’s query patterns\{38\} or adding perturbation according to specific inference results\{28\}. The latter type can be implemented by the server before the inference service, allowing it to collaborate with privacy-preserving inference protocols without compromising the client’s input privacy. These strategies, on the other hand, mitigate model extraction attacks by sacrificing model accuracy, necessitating a trade-off between model utility and defense effectiveness.

Since the query samples and the final inference results are protected from the server in secure inference, the passive defense methods, without requiring additional knowledge about the client’s query samples or output labels, are well compatible with our scheme. We adopt a passive defense method proposed in\{29\} because it mitigates the model extraction attacks by perturbing the server’s model network. As the attacker essentially approximates the loss hypersurface to find parameters for the stolen model with the minimum loss value, the defense method leverages a new activation layer that manipulates the estimated loss surface by adding a small controllable perturbation and thus maximizes the loss of the stolen model while preserving the accuracy of the original model. Specifically, this defense method only perturbs the model’s final activation layer (e.g., softmax) that maps a vector to a number of probability values.

The Reverse Sigmoid perturbation $r(y^j_i)$ is as follows.

$$r(y^j_i) = \beta s(\gamma s^{-1}(y^j_i)) - 1/2),$$

where $y^j_i$ represents $i$-th dimension of probability vector $y^j$ for sample $x_i$, $s(\cdot)$ is a sigmoid function, $\gamma$ is a positive dataset and model specific convergence parameter, and $\beta$ is a positive magnitude parameter.

The reverse sigmoid perturbation only replace the final layer. Then the final perturbed probability value is calculated as follows.

$$\hat{y}^j_i = \alpha^j(y^j_i - \beta s(\gamma s^{-1}(y^j_i)) - 1/2))$$

where $\alpha^j$ is a sum-to-1 normalizer for $y^j$.

B. Privacy-Preserving Inference

The privacy-preserving neural network inference takes the query samples from the client and the model parameters from the server as inputs. Since Fusion is a general compiler and utilizes any efficient semi-honest inference protocol as the underlying building block, we do not give a detailed description of the cryptographic techniques. Besides, due to the complicated computation in inference, privacy-preserving inference generally adopts more than one type of cryptographic technique. Thus, we give a high-level summary of several popular semi-honest privacy-preserving inference frameworks.

To implement privacy-preserving inference in the non-outsourcing scenario, secure 2PC is adopted. 2PC is a rapidly developing area of cryptography and allows two parties to perform a distributed computation while protecting the privacy of their inputs and the correctness of the outputs. Due to the complexity of operation types (e.g., nonlinear maximum, matrix multiplication) and plenty of conversions between these operations in neural networks, it poses a efficiency challenge. Hybrid 2PC inference frameworks\{24, 32, 36\} using multiple cryptographic building blocks (including Yao’s GC, secret-sharing, and HE) are very popular due to its efficiency. Secret sharing based schemes can achieve good efficiency in low latency network. HE-based scheme requires constant and less communication rounds, and high latency network has fewer impact on these schemes.

Hybrid inference schemes can be divided into two types according to the cryptographic techniques (e.g., secret sharing and homomorphic encryption) used to perform the linear operations. For instance, some schemes\{10, 26, 41, 44\} use secret sharing to perform linear layers. Secret sharing based schemes are computationally inexpensive but have non-constant communication rounds. In secret sharing based inference protocols, linear layers are typically performed under Arithmetic sharing and non-linear layers under Boolean sharing. Thus, secure conversion protocols between the Arithmetic sharing and Boolean sharing are required. Naturally, the intermediate results of each layer that each party obtains are random Arithmetic or Boolean shares.

Some other schemes\{18, 24, 36, 45\} schemes use homomorphic encryption to perform linear layers, which require constant number of communication and high computation overhead. The non-linear operations can be performed by using GC or OT-based comparison protocol\{45\}. The multiplication computation results completed by the HE are additive secret shares that are used as the inputs for the next non-linear layer. Similarly, the outputs of the non-linear layer are also secret shares of the computation results on non-linear function.

Cheetah: Cheetah\{18\} is an efficient semi-honest secure inference scheme. As Fusion used Cheetah as the underlying semi-honest 2PC inference scheme for instantiation, we here give some high-level description about Cheetah and refer to\{18\} for more cryptographic technique details. Specifically, Cheetah uses two lattice-based homomorphic encryptions\{6\}. 
i.e., learning with errors (LWE) and its ring variant (ring-LWE) to perform secure linear layers (e.g., fully connection, convolution, and batch normalization layers) and OT-based millionaire protocol for non-linear layer (e.g., ReLU function and Max pooling). They achieve performance optimizations based on the insightful observation that matrix multiplication results can be represented as the coefficients in specific positions of polynomial multiplication which can be efficiently performed using ring-LWE. When performing inference computations, conversions back and forth between the linear and non-linear layers are required. Notice that in each layer, the intermediate results that each party obtains are additive secret shares of the computation results, e.g., \([y]_L + [y]_C = f(\cdot, \cdot)\).

IV. PROBLEM STATEMENT

In privacy-preserving neural network inference, the server and the client jointly perform secure 2PC inference computations (described in section III-A) with their private inputs, i.e., model parameters and query samples, respectively. When the secure inference computations are accomplished, the inference results are only revealed to the client. In this section, we describe the system model, threat model, and security requirements.

A. System Model

There are two entities in the system model shown in Fig. 1.

- **Server:** The server owns a well-trained (in terms of accuracy) model for a specific task and provides an inference service to the client. The server aims to convince the client that the inference results are correctly computed by using low-quality model as input or deviating from the inference protocol.
- **Client:** The client honestly follows the protocol but trying to infer more information about the server’s input. The client may perform the black-box model extraction attack by using query samples and corresponding inference results.

Note that this attack only requires the client’s query samples and corresponding inference results (which are revealed to the client) to steal the server’s model. As it expects accurate inference results even if when performing the extraction attack, this setting is reasonable that the client will honestly follow the protocol.

B. Threat Model

In this paper, we design Fusion for a threat model called server-malicious security (similar as [13, 29, 51]) in which the server can deviate from the protocol while the client is semi-honest and follows the protocol specification but attempts to infer additional information.

C. Security goals

Due to the privacy preservation and result reliability requirements, the secure inference scheme should meet the following security goals: (1) verification of model accuracy, (2) privacy preservation, and (3) inference computation correctness. This type of security requirements are ensured by cryptographic techniques and will be proved in the simulation paradigm [50]. Furthermore, note that cryptographic techniques cannot prevent the model privacy leakage caused by the black-box model extraction attacks which only require input/output pairs. To address this issue, it is required to mitigate the attacks from an honest-but-curious client. This security is evaluated by the defense effectiveness, e.g., reducing the accuracy of the client’s stolen model.

1) **Cryptographic Security Requirements:** We provide security in the simulation paradigm. We consider a two-party hybrid model where parties both interact with each other and have access to ideal functionalities. Assume a two-party hybrid model protocol \(\pi\) uses ideal calls to ideal functionalities \(f_1, \cdots, f_{p(n)}\). Let \(\rho_1, \cdots, \rho_{p(n)}\) be protocols that securely compute \(f_1, \cdots, f_{p(n)}\) respectively. The composition theorem [3] states that if \(\pi\) securely computes the functionality \(g\) in the \((f_1, \cdots, f_{p(n)})\)-hybrid model, then \(\pi^{\rho_1, \cdots, \rho_{p(n)}}\) in which the ideal functionalities are substituted by the secure sub-protocols securely computes \(g\) in the real model.

**Definition 1:** A protocol \(\Pi_{\text{Fusion}}\) between a server having model parameters \(M\) which satisfy the accuracy threshold \(\delta\) and a client having a dataset \(X = (x_1, \cdots, x_n)\) as inputs securely achieves a secure inference functionality \(F_{\text{Fusion}}\) against a malicious server and a semi-honest client if it satisfies the following properties:

- **Model Accuracy.** The accuracy (e.g., \(\eta\)) of the model that the server uses as input should meet the requirement (e.g., \(\delta\)). Without compromising the privacy of the server’s model, the model accuracy \(\eta\) is calculated. If \(\eta \geq \delta\), the verification of model accuracy passes.
- **Computation Correctness.** In an execution of \(\Pi_{\text{Fusion}}\), the probability that the client’s output on every input vector \(x_i\) is not the correct inference result \(M(x_i)\) is negligible in security parameter \(\lambda\).
• Security.
  - Malicious server security. The view of the server during a real execution of protocol $\Pi_{\text{Fusion}}$ is denoted by $\text{View}_{S}^{\Pi_{\text{Fusion}}}$. For any server $S$, there exists a probabilistic polynomial-time simulator $\text{Sim}_S$ such that for any input $M$ of the server and $x_i$ of the client, we have:
  \[
  \text{View}_{S}^{\Pi_{\text{Fusion}}} \approx_c \text{Sim}_S(M)
  \]
  That is, $\text{Sim}_S$ can simulate a computationally indistinguishable view of the malicious server without knowing the client’s private inputs and inference results.
  - Semi-honest client security. The view of the client during a real execution of protocol $\Pi_{\text{Fusion}}$ is denoted by $\text{View}_{C}^{\Pi_{\text{Fusion}}}$. For any client $C$, there exists a probabilistic polynomial-time simulator $\text{Sim}_C$ such that for any input $M$ of the server and $x_i$ of the client, we have:
  \[
  \text{View}_{C}^{\Pi_{\text{Fusion}}} \approx_c \text{Sim}_C(x_i, M(x_i))
  \]
  That is, $\text{Sim}_C$ can simulate a computationally indistinguishable view of the semi-honest client without knowing the server’s model parameters.

2) Security requirement against the model extraction attacks: We further consider a strong threat model in which a curious client could perform black-box model extraction attacks. As a curious client (e.g., attempting to extract the server’s model) desires to obtain correct inference results on its query samples, it will honestly follow the protocol. This is consistent with the security assumption that the client is semi-honest, i.e., honestly following the inference protocol but trying to infer more information.

The goals of model extraction attacks can be divided into two objectives, i.e., accuracy, which measures the extracted model’s correctness on test samples, and fidelity, which measures the agreement between the extracted model and original model on any point. We measure the goal of client’s attack by accuracy because it is natural that the clients steal the model for use.

Since the server cannot prevent the curious client from obtaining query samples and inference results that are used by the client to extract the model, it is difficult to detect and completely prevent the attacks. Informally, the goal of the defense against the client’s model extraction attacks is to reduce the accuracy of the stolen model while maintaining the utility of the server’s defense model.

Formally, there are two metrics \[38\] to measure the effectiveness of the defense.

• Non-replicability. The non-replicability is measured in terms of the accuracy of the client’s stolen model. That is, the accuracy of the client’s stolen model should be far lower than that of the server’s model.
• Utility. The utility is measured in terms of the accuracy of the server’s model. The defense method should have little impact on the server’s model to maintain its utility.

V. THE FUSION SCHEME

In this section, we propose Fusion (shown in Fig. 3), a secure inference scheme that is secure against a malicious server and an honest-but-curious client.
Inference network, the defense model’s utility will not be affected compared to the trained model without defense, which is shown in Subsection VII-D. The server uses the defense model to provide inference service while mitigating the client’s model extraction attacks.

**B. Mixed Dataset Preparation**

Before the inference computations begin, the client locally prepares the **mixed dataset**. Specifically, the client owns a number (e.g., \( R \)) of query samples, duplicates each of its query samples (e.g., \( B \) copies), and prepares \( T \) public samples for testing the model accuracy. At last, the client uses a randomly chosen permutation to shuffle all copies of the query samples and public samples together.

**Searching optimized \( T \) and \( B \):** The statistical parameter is denoted by \( \lambda \). The security requirement is to guarantee that the server succeeds in cheating with a probability at most \( 2^{-\lambda} \). Amortized computation and communication overhead per query sample is proportional to the number of public samples and the number of copies for each query sample. We aim to find the optimal \( T \) and \( B \) that minimize the amortized cost while satisfying the security constraints. Next, we turn to how to pick the concrete numbers of \( B \) and \( R \) that satisfy the security requirement while also minimizing computation costs.

The server can succeed in cheating when (1) the calculated model accuracy by using inference results on the \( T \) public samples meets the client’s requirement, and (2) the inference results of all copies for every query sample are consistent and the inference results of some query samples are incorrect. To simplify the problem, we assume that the server knows \( T \) and \( B \). Assume that the server wants to corrupt \( i \) query samples by providing \( iB \) incorrect-but-consistent inference results for all copies of every query sample. If the server wants to pass the check of model accuracy, it should use the well-trained model as input for the \( T \) public samples. Let \( E_T \) denote the event in which the server uses a well-trained model to correctly perform the inference computations on all \( T \) public samples. The probability \( \Pr[E_T] \) that event \( E_T \) happens is as follows.

\[
\Pr[E_T] = \frac{\binom{RB + T - iB}{T}}{\binom{RB + T}{T}} = \frac{(RB + T - iB)! (RB)!}{(RB - iB)! (RB + T)!}.
\]

Let \( E_B \) denote the event in which the \( iB \) incorrect inference results chosen by the server are exactly the incorrect-but-consistent results for \( i \) query samples. There are \( (RB)! \) ways to permute the samples. If the server wants to cheat successfully, it should provide incorrect inference results for \( iB \) copies of the \( i \) query samples, and use the well-trained model to perform correct inference computations for remaining samples. The probability \( \Pr[E_B] \) that event \( E_B \) happens is as follows.

\[
\Pr[E_B] = \frac{\binom{R}{i} (iB)! (RB - iB)!}{(RB)!}.
\]

Combining Eq. 3 and Eq. 4, the probability \( \Pr_{success} \) that the server succeeds in cheating is as follows.

\[
\Pr_{success} = \Pr[E_T \land E_B] = \Pr[E_T] \times \Pr[E_B] = \left( \frac{R}{i} \right) \left( \frac{RB + T}{iB} \right)^{-1}.
\]

If \( T \geq B \), \( \left( \frac{R}{i} \right) \left( \frac{RB + T}{iB} \right)^{-1} \leq R \left( \frac{RB + T}{iB} \right)^{-1} \), which is proved in Section VII.

Therefore, the parameter optimization problem can be expressed as the follows.

\[
\arg \min_{B,T} \frac{RB + T}{R}.
\]
Protocol $\Pi_{\text{Fusion}}$ for batched secure inference

**Input:** The client inputs a mixed dataset $X = (x_1, \cdots , x_{RB+T})$ including $T$ public samples and $R$ query samples (each with $B$ copies), the server inputs model $M$ for a specific inference task, and an accuracy threshold $\delta$ is set.

**Output:** The inference results $M(x_i)$ for all samples $i \in [RB + T]$ in the mixed dataset.

1) **Defense against model extraction attacks.**

   The server trains a defense model by using reverse sigmoid perturbation and uses the defense model to provide inference service.

2) **Mixed dataset preparation (only the client).**

   a) Searches optimal $T$ and $B$ that minimize the amortized cost.
   
   b) Duplicates each query sample $B$ copies and uses a randomly chosen permutation to shuffle all $RB + T$ samples.

3) **Privacy-preserving inference computations.**

   For all $RB + T$ samples in the mixed dataset, the client and the server jointly perform the semi-honest privacy-preserving inference computations. After the inference computations are completed, the inference results $M(x_i), i \in [RB + T]$ are only revealed to the client.

4) **Model accuracy and computation correctness verification (only the client):**

   a) Calculates the **model accuracy** $\eta$ as follows.

   $\eta = \frac{\text{Number of correct inference on } T \text{ public samples}}{T}$.

   If $\eta < \delta$, the client aborts.

   b) Verifies the **computation correctness** by checking the consistency of the inference results on $B$ copies of each query sample. If any of the $B$ copies of a query sample are inconsistent, the client aborts.

---

subject to

\[ T \geq \beta, \quad \text{(7)} \]

\[ \Pr_{\text{success}} \leq 2^{-\lambda}, \quad \text{(8)} \]

where $\beta$ is the least number of public samples to ensure the reliability of model accuracy.

Given the security parameter $\lambda$, the threshold of public samples $\beta$, and the number of query samples $R$, the goal is to find the optimal $B$ and $T$ that lead to the least amortized cost. When both $E_T$ and $E_B$ succeed, the server can cheat successfully. The security requirement states that $\Pr_{\text{success}}$ should be no more than $2^{-\lambda}$ for every choice of $i$ by the server and of $T$ and $B$ by the client.

Based on these constraints and observations, we design a search algorithm shown in Fig. 4 to solve the parameter optimization problem. The basic idea is summarized as follows. For every $B = 2, 3, \cdots$, such that the smallest $T$ ($T \geq \beta$) satisfying $\Pr_{\text{success}} \leq 2^{-\lambda}$ is found. At last, the pair $(B, T)$ that minimizes $\frac{RB + T}{R}$ is chosen to be the most optimized parameter pair.

**C. Privacy-Preserving Inference Scheme**

In this phase, the client requests inference services from the server using the private mixed dataset as input and jointly performs the secure inference computations with the server. The inference computations are implemented using semi-honest privacy-preserving inference protocols. Fusion provides flexibility in choosing the 2PC based privacy-preserving inference scheme. For every sample in the mixed dataset, the semi-honest 2PC based inference protocol is invoked to obtain the inference result. Finally, the inference results on all samples in the mixed dataset are only revealed to the client. At the end of this phase, the correctness of the inference results is not guaranteed and will be checked in the next phase.

**D. Model Accuracy and Computation Correctness Verification**

When obtaining inference results on the mixed dataset, the client checks the **model accuracy** and **computation correctness** as follows.

1) **Model accuracy.** The client computes the model accuracy $\eta$ as follows.

   \[ \eta = \frac{\text{Number of correct inference on } T \text{ public samples}}{T}. \]

2) **Computation correctness.** The client verifies the computation correctness by checking the consistency of the inference results on $B$ copies of each query sample. If any of the $B$ copies of a query sample are inconsistent, it is considered that the server attempted to deceive the client by giving incorrect inference results.

   The client accepts the inference results if both checks pass, otherwise, it aborts.

---

**VI. Security Analysis**

In this section, we give the security analysis of Fusion.
Theorem 1: Assuming the existence of $\Pi_{\text{PPML}}^{\text{semi}}$ that securely realizes inference functionality $F_{\text{PPML}}^{\text{semi}}$ under the semi-honest security, the protocol $\Pi_{\text{Fusion}}$ for achieving secure inference securely realizes the ideal functionality $F_{\text{Fusion}}$ (with abort) in the presence of the malicious server in the $F_{\text{PPML}}^{\text{semi}}$-hybrid model.

We need to prove that there exists a simulator who can generate a view in the real world that is indistinguishable from the view in the real world for the adversary. We first consider that the client is corrupted by a semi-honest adversary.

Proof: Client is corrupted. As described Definition 1, we have to prove that there exists a simulator $\tilde{S}$ that can simulate the view that is indistinguishable from the view in the real world for the adversary. Since the client is semi-honest and acts the identical way as the client in $F_{\text{PPML}}^{\text{semi}}$ except for additional local checks, there exists a simulator $\tilde{S}$ that can simulate the indistinguishable view by invoking the simulator of $F_{\text{PPML}}^{\text{semi}}$.

Server is corrupted. We then consider that the server is corrupted by a static and active adversary. When the server is corrupted by a malicious adversary who can deviate from the protocol, the final output of the honest client may be incorrect and causes abort in both worlds. Then we construct a simulator $\tilde{S}$ that simulates the view in the ideal world. We will show that the view in the ideal world is statistically indistinguishable from the view in the real world. In step 3 of $\Pi_{\text{Fusion}}$, we let simulator $\tilde{S}$ invoke the simulator of $F_{\text{PPML}}^{\text{semi}}$ and outputs whatever it outputs. The intermediate results that the corrupted server obtains are random secret shares of the computation results. Next in step 4 of $\Pi_{\text{Fusion}}$, the simulator $\tilde{S}$ works as follows.

Case 0: If the adversary does not cheat throughout the protocol, there are two probable results. If the model accuracy does not satisfy the threshold, the simulator will sends abort to the functionality while in the real world the client rejects the results and aborts. If the model accuracy satisfies the threshold, the functionality sends inference results to the client while in the real world the clients accepts the results. The view in both the ideal and the real worlds is identical.

Case 1: If the adversary cheats by using inconsistent model parameters for multiple query samples or deviating from the protocol, the simulator will detect and sends abort to the functionality. In the real world, if the server cheats as above, it will be caught with overwhelming probability.

The adversary can cheat successfully if all copies of the same query sample obtain incorrect-but-consistent inference results. The mix-and-check method ensures that the cheating probability can be negligibly small when appropriately choosing the parameters $B$ and $T$.

We begin by defining the following mix-and-check game, which is equivalent to our protocol $\Pi_{\text{Fusion}}$. The server wins the game if he chooses some of the query samples and provides incorrect inference results for them without being caught by the client. We need to prove that the probability of the server winning the game is negligible, that is, the server cannot distinguish every two samples in the mixed dataset. If the game’s output is 1, the malicious server wins the game and cheats the client successfully. The probability $Pr_{\text{success}}$ that the server succeeds in cheating equals $Pr_{\text{Game}}(S, C, R, B, T) = 1$.

Definition 2: The probability that the server $S$ wins the game is negligible by choosing appropriate $R$, $B$, and $T$.

The game $\text{Game}(S, C, R, B, T)$ proceeds as follows.

1) The client $C$ prepares the mixed dataset containing $RB + T$ samples and uses them as inputs for privacy-preserving inference computations.

2) The server $S$ selects $iB$ samples ($i$ is the number of query samples $S$ chooses to fool) in the mixed dataset and returns incorrect inference results for each of them. The inference results for the remaining samples are computed correctly using the well-trained model parameters.

3) The output of the game is 1 if there are $i$ query samples such that inference results for each of them and corresponding copies are incorrect-but-consistent, while inference results for remaining $R - i$ query samples and their copies are obtained by correctly performing the inference computations using well-trained model parameters.

Claim 1: If $T \geq B$, then for every adversary $S$, it holds that

$$Pr_{\text{Game}}(S, C, R, B, T) = 1 \leq R\left(\frac{RB + T}{B}\right)^{-1}.$$  

Proof: We need to show that for every $1 \leq i \leq R$,

$$\left(\frac{R}{i}\right)\left(\frac{RB + T}{iB}\right)^{-1} \leq R\left(\frac{RB + T}{B}\right)^{-1}. \quad (9)$$

At first, it can be observed that when $i = 1$, the left side of the inequality equals the right side, and thus the equation holds. Next, assume that $i \geq 2$, it suffices to show that:

$$\left(\frac{R}{i}\right)\left(\frac{RB + T}{iB}\right)^{-1} \leq R\left(\frac{RB + T}{B}\right)^{-1}.$$  

It is equivalent to proving that:

$$\left(\frac{R}{i}\right)\frac{(RB + T - iB)!}{(RB + T)!} \leq \frac{B!(RB + T - B)!}{(RB + T)!},$$

which can be represented as:
\[ (R_i^{|iB|})! \leq \frac{(RB + T - B)!}{(RB + T - iB)!}. \]

By multiplying both sides with \( (iB)! \), the above inequality can be transferred to
\[ (R_i^{|iB|} - 1) \leq \frac{(RB + T - B)!}{(RB + T - iB)!}. \]

Considering the assumption that \( T \geq B \) and thus \( RB \leq (RB + T - B) \), it suffices to prove that
\[ R_i^{|iB|} - 1 \leq \frac{RB}{iB - B}. \]

To prove that the above Eq. (10) holds, we can consider the both sides of the inequality as following process.

1) The left side \( \left( R_i^{|iB|} - 1 \right) \) can represent the process:
choosing \( i \) query samples among \( R \) query samples, then
choosing \( iB - B \) samples from \( iB \) copies of the selected
\( i \) query samples, and using false model parameters to provide inference results for the \( iB - B \) samples.

2) The right side \( \frac{RB}{iB - B} \) can represent the process:
choosing \( iB - B \) samples from \( RB \) samples.

The above two processes both end with choosing \( iB - B \) samples out of \( RB \) samples. Since there is no restriction on the selection of the right process, the number of choices in the right process is strictly larger than that in the left process. It is sufficient to conclude that the inequality Eq. (10) holds.

**Claim 2:** In real execution, if the parameter \( B \) and \( T \) are properly chosen as in \( \Pi_{\text{Search}} \), then the client aborts with probability at least \( 2^{-\lambda} \).

When \( B \) and \( T \) are properly chosen as in \( \Pi_{\text{Search}} \), it ensures that \( R_i^{|iB|} - 1 \leq 2^{-\lambda} \), then the probability \( \Pr[\text{Game}(S,C,R,B,T) = 1] \leq 2^{-\lambda} \). That is, if the server cheats in the real protocol, the client will detect and abort with probability at least \( 1 - 2^{-\lambda} \), while the simulator will definitely detect the server’s cheating behavior and aborts in the ideal world. Thus, the view in both the ideal and real worlds is statistically close.

To conclude, in all cases, the view in both the ideal and real worlds is computationally indistinguishable. Thus, the protocol \( \Pi_{\text{Fusion}} \) securely realizes the ideal functionality in the \( \mathcal{F}^{\text{PPML}} \)-hybrid model against the malicious server. This completes the proof.

**VII. PERFORMANCE EVALUATION**

We empirically validate the efficiency of Fusion. At first, we show that our maliciously secure scheme are more efficient than maliciously secure deep learning LevioSA [15] in Table II, then, we test the impact of the existence of defense/attack on Fusion’s performance and show the result in Table IV and V. Then we show the scalability of Fusion on practical ImageNet-scale benchmarks and comparison with semi-honest inference scheme in Table VI. Next, the effectiveness of Fusion’s defense is shown in Fig. 6. Finally, experimental results on real-world medical datasets are shown in Table VII.

**Experimental Setup.** For comparison experiments in Table we set up completely identical experimental configurations to those used by LevioSA. There are two servers running Ubuntu 16.08 with 2.3 GHz Intel Xeon E5 Broadwell Processors and 244GB RAM. We set the latency as 40 ms which is higher than the latency between two Amazon EC2 machines located in Ohio and Virginia used in LevioSA. Other experiments were carried on Intel Xeon E5-2630 v3 @ 2.40GHz with 128GB of RAM and Intel Xeon CPU E5-2680 v4 @ 2.40GHz with 256GB of RAM. The bandwidth between the machines were 10 Gbps and 44 MBps in the LAN and the WAN setting respectively, and the echo latency were 0.4 ms and 40 ms.

**A. Comparison and Improvements**

We compare our scheme with maliciously secure LevioSA [15]. We employed the same neural network framework as LevioSA, which consists of a four-layer deep neural network (DNN) with three hidden, fully connected layers with 2000 neurons, quadratic activations, and the final layer is fully connected with 183 output neurons. We also trained the network on the TIMIT dataset [12] as they used. In the inference phase, the mixed dataset is prepared by combining 1,845 speech samples (each with 5 copies) and 100 speech samples (used as public samples).

The experimental results of communication and runtime between LevioSA and Fusion are shown in Table III. It indicates that Fusion has 48.06× less runtime and uses 30.90× less communication compared to LevioSA. Furthermore, LevioSA does not guarantee the verification of the server’s model accuracy and cannot defend against the client’s model extraction attacks, which are achieved in our scheme.

| Scheme   | Comm. (GiB) | Runtime (min) |
|----------|------------|---------------|
| LevioSA  | 20.7       | 34.6          |
| Fusion   | 0.67       | 0.72          |

TABLE II: Comparison of communication and runtime between LevioSA [15] and our implementation of Fusion using Cheetah.

**B. Impact of the existence of defense/attack on Fusion’s performance**

To simply the problem, we set the number of copies \( B \) from ten to three, and search \( (R,T) \) that satisfies \( \Pr_{\text{success}} \leq 2^{-\lambda} \). For each query sample, we test the B for statistical parameter \( \lambda = 40 \) and \( \beta = 100 \). The results are shown in Table III. Then we let the client use these concrete numbers of \( R \) and \( T \) to generate the mixed dataset that ensures the statistical security.

To show the performance of Fusion thoroughly, we test the communication and runtime on two datasets (MNIST in Table IV and CIFAR-10 in Table V) in a variety of settings, i.e., the presence of the server’ defense, the presence of the
TABLE III: Finding concrete \((R, T)\) with different \(B\) that satisfies the security parameter.

| Number of samples | MNIST |          |          |          |          |          |
|-------------------|-------|----------|----------|----------|----------|----------|
|                   | ND, NA | WA, NA   | WD, A1   | WD, A2   |
|                   | Comm. | Runtime | Comm. | Runtime | Comm. | Runtime | Comm. | Runtime |
| \(2^1 \times 9 + 100 = 118\) | 0.1373 | 0.0027 | 0.1376 | 0.0027 | 0.1377 | 0.0027 | 0.1374 | 0.0027 |
| \(2^3 \times 8 + 100 = 164\) | 0.1907 | 0.0039 | 0.1914 | 0.0039 | 0.1916 | 0.0039 | 0.1911 | 0.0039 |
| \(2^5 \times 7 + 100 = 324\) | 0.3765 | 0.0073 | 0.3782 | 0.0073 | 0.3785 | 0.0073 | 0.3774 | 0.0073 |
| \(2^7 \times 6 + 100 = 868\) | 1.0104 | 0.0198 | 1.0133 | 0.0198 | 1.0159 | 0.0198 | 1.0118 | 0.0198 |
| \(2^9 \times 5 + 100 = 2,660\) | 13.0974 | 0.0605 | 3.1048 | 0.0605 | 3.1084 | 0.0605 | 3.1011 | 0.0605 |
| \(2^{13} \times 4 + 100 = 32,868\) | 38.2043 | 0.7428 | 38.3905 | 0.7429 | 38.4397 | 0.7429 | 38.3467 | 0.7429 |
| \(2^{19} \times 3 + 100 = 1,572,964\) | 1,828.7964 | 35.7579 | 1,832.7539 | 35.7586 | 1,836.9962 | 35.7586 | 1,831.8406 | 35.7586 |

TABLE IV: Communication (MiB) and runtime (s) of Fusion instantiated using Cheetah on MNIST. NA denotes no attack while WA denotes with attack. Similarly, ND and WD represent no defense and with defense respectively. A1 and A2 represent attack 1 and attack 2, respectively.

| Number of samples | CIFAR-10 |          |          |          |          |          |
|-------------------|----------|----------|----------|----------|----------|----------|
|                   | ND, NA | WA, NA   | WD, A1   | WD, A2   |
|                   | Comm. | Runtime | Comm. | Runtime | Comm. | Runtime | Comm. | Runtime |
| \(2^1 \times 9 + 100 = 118\) | 0.1682 | 0.0033 | 0.1691 | 0.0033 | 0.1693 | 0.0033 | 0.1686 | 0.0033 |
| \(2^3 \times 8 + 100 = 164\) | 0.2336 | 0.0046 | 0.2350 | 0.0046 | 0.2352 | 0.0046 | 0.2342 | 0.0046 |
| \(2^5 \times 7 + 100 = 324\) | 0.4617 | 0.0091 | 0.4637 | 0.0091 | 0.4648 | 0.0091 | 0.4628 | 0.0091 |
| \(2^7 \times 6 + 100 = 868\) | 1.2354 | 0.0243 | 1.2436 | 0.0243 | 1.2445 | 0.0243 | 1.2397 | 0.0243 |
| \(2^9 \times 5 + 100 = 2,660\) | 13.7921 | 0.0745 | 3.8084 | 0.0745 | 3.8127 | 0.0745 | 3.7994 | 0.0745 |
| \(2^{13} \times 4 + 100 = 32,868\) | 46.8642 | 0.9202 | 47.0841 | 0.9203 | 47.3280 | 0.9203 | 46.9062 | 0.9203 |
| \(2^{19} \times 3 + 100 = 1,572,964\) | 2,247.9946 | 44.1955 | 2,256.4829 | 44.1961 | 2,264.8305 | 44.1961 | 2,251.0891 | 44.1961 |

TABLE V: Communication (MiB) and runtime (s) of Fusion instantiated using Cheetah on CIFAR-10. NA denotes no attack while WA denotes with attack. Similarly, ND and WD represent no defense and with defense respectively. A1 and A2 represent attack 1 and attack 2, respectively.

Impact of the existence of defense. To investigate the impact of the defense on efficiency, we conduct experiments in which the only variable is the presence of the defense while all other settings remain the same. As Fig. 5. (a) and (d) show, the existence of defense has little impact on the running time and communication when compared to without adopting defense. When the total number of samples is 1,572,964, the Fusion with defense increases only 3.956/8.488 MiB communication and 0.0007/0.0006 s runtime overhead compared to that without defense using dataset MNIST and CIFAR-10 respectively, which is negligible. The reason for this is that the defense method only changes the final activation layer and adds very little computation compared to the overall computation.

Impact of the existence of attack. As Fig. 5. (b) and (e) show, the existence of the client’s attack does not affect the running time and communication. Similarly, when the client uses different attack approaches, as shown in Fig. 5. (b) and (e), the running time and communication remain nearly the same.
same. The reason for this is that the client only exploits the attack after receiving the inference results, so it has no effect on the privacy-preserving inference computations.

### C. Evaluation on practical DNN ResNet50

To show the scalability of Fusion, we also conduct empirical evaluation of maliciously secure inference on ImageNet-scale deep neural networks (DNN) ResNet50 [16]. We train a model and perform inference on ResNet50 using an image set [1]. We evaluated the same trained ResNet50 model while using different privacy-preserving 2PC frameworks, i.e., maliciously secure Fusion and semi-honest secure CryptFlow2, and the experimental result is shown in Table VI. Since the number of batched query samples \( R \) affects the amortized cost per query sample, we choose \( R = 8, 32, 128, \) and \( 512 \), and the corresponding number of copies for each query sample varies from 8 to 5. The larger the number of the total query sample is (which can reduce the number of copies for each query sample), the lower the amortized cost. It shows that as \( R \) increases and corresponding \( B \) decreases, the amortized cost for obtaining inference results per query sample decreases in Fusion. When \( R \) is larger than 32 and \( B \) is smaller than 7, the amortized communication of maliciously secure Fusion is less than that of semi-honest CryptFlow2. When \( R = 512 \) and \( R = 5 \) that ensures statistical security, Fusion costs 1.30× runtime and is 1.18× faster in the LAN setting and the WAN setting respectively, and has 2.64× less communication than that of CryptFlow2.

![Figure 5: The impacts of different settings on the performance is depicted in this figure. M and C represent MNIST and CIFAR-10 datasets, respectively. NA denotes no attack while WA denotes with attack. Similarly, ND and WD represent no defense and with defense respectively. A1 and A2 represent attack 1 and attack 2, respectively.](image)

| Scheme               | Comm. (Mb) | Runtime (LAN) | Runtime (WAN) |
|----------------------|------------|---------------|---------------|
| Fusion \((R = 8, B = 8)\) | 39.921     | 20.410        | 34.241        |
| Fusion \((R = 32, B = 7)\) | 19.714     | 10.082        | 16.912        |
| Fusion \((R = 128, B = 6)\) | 13.205     | 6.750         | 11.326        |
| Fusion \((R = 512, B = 5)\) | 10.117     | 5.173         | 8.678         |
| CryptFlow2 [45] \((SCI)\) | 26.742     | 3.988         | 10.204        |

**TABLE VI: Performance on ResNet50 and ImageNet scale benchmarks.** It shows amortized communication (GiB) and runtime (minutes) per query sample when \( R \) and \( B \) vary. The number of test public samples \( T = 100 \) for all batched queries.

### D. Effectiveness of defense against model extraction attacks

We now demonstrate the effectiveness of Fusion’s defense against model extraction attacks. The accuracy of the defense model and the stolen model is shown in Fig. 6 when the datasets, the number of samples, and stealing approaches vary.

The experimental results for both datasets show that when the number of samples reaches 2,660, the accuracy of the stolen model increases slowly and then remains steady. Fusion slightly degrades the accuracy of the server’s model when compared to that without defense, namely, the utility of the defense model is maintained. For instance, when using the MNIST dataset, the server’s model with defense is 96.78%, while the model without defense is 98.53%. When the dataset is CIFAR-10, the server’s model with defense is 75.34%, while the model without defense is 76.31%.
We use two attack approaches, i.e., attack 1 [40] and attack 2 [4], to test the defense effectiveness of Fusion. When using the MNIST dataset, the accuracy of the client’s stolen model decreases by 36.1-42.7% (attack 1), while the accuracy decreases by 33.1-39.8% (attack 2) as the number of query samples varies from 118 to 1,572,964. For instance, when the number of samples is 1,572,964, the accuracy of the client’s stolen model drops from 95.81%, 96.25% (without defense) to 58.42%, 63.15% (with defense) under attack 1 and attack 2, respectively.

When using the CIFAR-10 dataset and defense method, the accuracy of the client’s stolen model decreases by 33.1-35.1% using attack approach 1, while the accuracy decreases by 24.2-29.8% using attack approach 2 as the number of query samples varies from 118 to 1,572,964. When the number of samples is 1,572,964, for example, the accuracy of the client’s stolen model drops from 71.9%, 72.88% (without defense) to 40.62%, 48.63% (with defense) under attack 1 and attack 2, respectively. To sum up, the accuracy of client’s stolen model is poor and difficult to implement, indicating the non-replicability of Fusion.

E. Evaluation on Medical Datasets

Since medical data is high-sensitive and has strong privacy preservation requirements, medical data analysis is one of the most important applications of secure inference. To show the applicability of maliciously secure inference in real-world medical datasets, we evaluated experiments on four publicly available healthcare datasets. We used the same DNN network as in Subsection VII-A to train these models and perform inference. The experimental results are shown in Table VII. BC-TCGA [54] and GSE2034 [57] are gene expression profiles, and PneumoniaMNIST [52] and DermaMNIST data [54] are X-Ray images and dermatoscopic images respectively. Take the BC-TCGA dataset as an example, it consists of 17,814 genes (features) and costs 14.579 ms/0.469 MiB in maliciously secure inference per query sample.

F. Further Discussion

In MLaaS, it may also suffer from model unfairness [35] or backdoor attacks [34] from a tricky server in real-world scenarios. Generally, these problems are covert even if without privacy preservation, and it requires complicated defense methods that are independent work of maliciously secure MPC. Nevertheless, there are some ways for the client to detect the unfair model [48] or the backdoor model [53] that would complement Fusion’s work. For instance, Fusion can use a detection strategy that involves no changes to the inference network and detects anomalous behaviour using tailored test samples.

VIII. CONCLUSION

Fusion can achieve security requirements including privacy and verification of model accuracy and computation correctness against a malicious server. It can be used as a general compiler by converting any semi-honest inference scheme into a maliciously secure one and thus can benefit from any efficient 2PC based inference scheme. Fusion maintains the...
server’s model utility while mitigating the model extraction attacks by decreasing the accuracy of the client’s stolen model.

In the future, we will focus on considering a stronger threat model that the client may give false reports about the inference results, and provide solutions for their reliability.

REFERENCES

[1] Imagenet-sample-images. https://github.com/EliSchwartz/imagenet-sample-images. 2020. 1000 images, one random image per image-net class. For easy visualization/exploration of classes.

[2] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J. Kusner, and Adrià Gascón. QUOTIENT: two-party secure neural network training and prediction. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, pages 1231–1247. ACM, 2019.

[3] Ran Canetti. Security and composition of multiparty cryptographic protocols. J. Cryptol., 13(1):143–202, 2000.

[4] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In Daniele Micciancio and Thomas Ristenpart, editors, Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, pages 189–218. Springer, 2020.

[5] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 395–412, 2019.

[6] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) lwe ciphertexts. In International Conference on Applied Cryptography and Network Security, pages 460–479, 2021.

[7] Zelei Cheng, Zuotian Li, Jiwei Zhang, and Shuhan Zhang. Differentially private machine learning model against model extraction attack. In 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybernetics (Cybernetics), pages 722–728. IEEE, 2020.

[8] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, pages 613–627. USENIX Association, 2017.

[9] Anders Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In 30th USENIX Security Symposium (USENIX Security 21), 2021.

[10] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In NDSS, 2015.

[11] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. Nature medicine, 25(1):24–29, 2019.

[12] MICHAEL FEKADU. Darpa timt acoustic-phonetic continuous speech. https://www.kaggle.com/datasets/mekadu/darpa-timt-acousticphonetic-continuous-speech/metadata. Accessed 2019.

[13] Boyuan Feng, Linanke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. Zen: An optimizing compiler for verifiable, zero-knowledge neural network inferences. Technical report, Cryptology ePrint Archive, Report 2021/687, 2021.

[14] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA., pages 4672–4681, 2017.

[15] Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. Leviosa: Lightweight secure arithmetic computation. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 327–344, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society, 2016.

[17] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. Privacy-preserving machine learning as a service. Proc. Priv. Enhancing Technol., 2018(3):123–142, 2018.

[18] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiasheng Ding. Cheetah: Lean and fast secure two-party deep neural network inference. In 31th USENIX Security Symposium (USENIX Security 22), 2022.

[19] Yuval Ishai, David Ben-Or, Amos Fiat, Tal Rabin, and Amir Yehuda Oren. Secure multiparty computation. In Proceedings of the 25th Annual ACM Symposium on Theory of Computing, pages 165–173. ACM, 1993.

[20] Norfolk, VA, USA, June 26-30, 2016. 2016. 1000 images, one random image per image-net class. For easy visualization/exploration of classes.

[21] hereby. The model is trained on the CIFAR-10 dataset, which contains 60,000 images, 50,000 of which are used for training and the remaining are used for testing.

[22] https://www.kaggle.com/datasets/mfekadu/darpa-timt-acousticphonetic-continuous-speech/metadata. Accessed 2019.

[23] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J. Kusner, and Adrià Gascón. QUOTIENT: two-party secure neural network training and prediction. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, pages 1231–1247. ACM, 2019.

[24] Ran Canetti. Security and composition of multiparty cryptographic protocols. J. Cryptol., 13(1):143–202, 2000.

[25] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In Daniele Micciancio and Thomas Ristenpart, editors, Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, pages 189–218. Springer, 2020.

[26] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 395–412, 2019.

[27] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) lwe ciphertexts. In International Conference on Applied Cryptography and Network Security, pages 460–479, 2021.

[28] Zelei Cheng, Zuotian Li, Jiwei Zhang, and Shuhan Zhang. Differentially private machine learning model against model extraction attack. In 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybernetics (Cybernetics), pages 722–728. IEEE, 2020.

[29] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, pages 613–627. USENIX Association, 2017.

[30] Anders Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In 30th USENIX Security Symposium (USENIX Security 21), 2021.

[31] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In NDSS, 2015.

[32] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. Nature medicine, 25(1):24–29, 2019.

[33] MICHAEL FEKADU. Darpa timt acoustic-phonetic continuous speech. https://www.kaggle.com/datasets/mekadu/darpa-timt-acousticphonetic-continuous-speech/metadata. Accessed 2019.

[34] Boyuan Feng, Linanke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. Zen: An optimizing compiler for verifiable, zero-knowledge neural network inferences. Technical report, Cryptology ePrint Archive, Report 2021/687, 2021.

[35] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA., pages 4672–4681, 2017.

[36] Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. Leviosa: Lightweight secure arithmetic computation. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 327–344, 2019.
[36] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Kaluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In 29th USENIX Security Symposium (USENIX Security 20), pages 2505–2522, 2020.

[37] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 35–52, 2018.

[38] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against DNN model stealing attacks. In 8th International Conference on Learning Representations, ICLR 2020. OpenReview.net, 2020.

[39] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish K. Shevade, and Vinod Ganapathy. Activethief: Model extraction using active learning and unannotated public data. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, pages 865–872. AAAI Press, 2020.

[40] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, pages 506–519. ACM, 2017.

[41] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. Aby2.0: Improved mixed-protocol secure two-party computation. In 30th USENIX Security Symposium (USENIX Security 21), 2021.

[42] Arpita Patra and Ajith Suresh. Blaze: blazing fast privacy-preserving machine learning. In NDSS, 2020.

[43] Rahul Rachuri and Ajith Suresh. Trident: Efficient 4pc framework for privacy preserving machine learning. In NDSS, 2020.

[44] Deevashwer Rathee, Mayank Rathee, Rahul Kranti Kiran Goli, Divya Gupta, Rahul Sharma, Nishanth Chandran, and Aseem Rastogi. Sirnn: A math library for secure rnn inference. In 2021 IEEE Symposium on Security and Privacy (SP), pages 1–18. IEEE, 2021.

[45] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 325–342, 2020.

[46] M Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. Xonn: Xnor-based oblivious deep neural network inference. In 28th USENIX Security Symposium (USENIX Security 19), pages 1501–1518, 2019.

[47] Mauro Ribeiro, Katarina Grodinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), pages 896–902. IEEE, 2015.

[48] Shahar Segal, Yossi Adi, Benny Pinkas, Carsten Baum, Chaya Ganesh, and Joseph Keshet. Fairness in the eyes of the data: Certifying machine-learning models. In Marion Fourcade, Benjamin Kuipers, Seth Lazar, and Deidre K. Mulligan, editors, AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021, pages 926–935. ACM, 2021.

[49] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In 2017 IEEE Symposium on Security and Privacy (SP), pages 3–18. IEEE, 2017.

[50] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In 25th USENIX Security Symposium, USENIX Security 16, pages 601–618. USENIX Association, 2016.

[51] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiaowei Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In 30th USENIX Security Symposium (USENIX Security 21), pages 501–518, 2021.

[52] Haozhe Xie, Jie Li, Tim Jatkoe, and Christos (2017) Hatzis. Gene expression profiles of breast cancer. Mendeley Data, 2017.

[53] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, and Bo Li. Detecting AI trojans using meta neural analysis. In 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24–27 May 2021, pages 103–120. IEEE, 2021.

[54] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. arXiv preprint arXiv:2110.14795, 2021.

[55] Andrew Chi-Chih Yao. How to generate and exchange secrets. In 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), pages 162–167. IEEE, 1986.

[56] Lingchen Zhao, Qian Wang, Cong Wang, Qi Li, Chao Shen, and Bo Feng. Veriml: Enabling integrity assurances and fair payments for machine learning as a service. IEEE Transactions on Parallel and Distributed Systems, 32(10):2524–2540, 2021.