MINISTRY OF EDUCATION AND TRAINING

VIETNAM ACADEMY OF SCIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

TRAN ANH TU

DEVELOPING CRYPTOGRAPHY-BASED SOLUTIONS TO ENHANCE SECURITY IN FEDERATED LEARNING

DOCTORAL THESIS IN COMPUTER SCIENCE

Hanoi – 2024

MINISTRY OF EDUCATION AND TRAINING

VIETNAM ACADEMY OF SCIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

TRAN ANH TU

DEVELOPING CRYPTOGRAPHY-BASED SOLUTIONS TO ENHANCE SECURITY IN FEDERATED LEARNING

Major: Computer Science
Code: 9480101

DOCTORAL THESIS IN COMPUTER SCIENCE

ACADEMIC SUPERVISORS
Supervisor 1: Assoc. Prof. LUONG THE DUNG
Supervisor 2: Prof. HUYNH VAN NAM

Hanoi – 2023

PLEDGE

I solemnly declare that this thesis is the product of my original research. It integrates insights from scholarly articles, papers presented at renowned international conferences, and books published by respected publishers. The findings and discussions presented here are unique and have not been previously published by other authors.

Hanoi, 2024

Ph.D. Student

Tran Anh Tu

ACKNOWLEDGEMENTS

Embarking on the journey of scientific discovery is both a challenging and rewarding experience, and this thesis stands as a significant milestone along that path. Throughout this endeavor, I have been fortunate to receive unwavering support and encouragement from many individuals, without whom this work would not have been possible.

First and foremost, I would like to express my sincere gratitude to my advisors, Associate Professor Luong The Dung and Professor Huynh Van Nam. Their guidance, expertise, and invaluable insights have been the cornerstone of this research. I am deeply thankful for the opportunity to learn from them, and for their constant support and encouragement, which has significantly shaped my academic journey.

I also wish to extend my heartfelt thanks to the faculty and staff of the Graduate University of Science and Technology, as well as my colleagues at the Academy of Cryptography Techniques. Their collaboration, support, and belief in my work have provided me with the motivation and inspiration necessary to push forward in this research. I am grateful for the stimulating academic environment they helped foster.

Lastly, I owe my deepest appreciation to my family—my spouse and children—whose love, patience, and understanding have been my greatest strength throughout this process. Their unwavering support allowed me to dedicate myself fully to this work, and for that, I am forever indebted.

Hanoi, 2024

Ph.D. Student

Tran Anh Tu

CONTENTS

LIST OF ABBREVIATIONS

API	Application Programming Interface
BNN	Binary Neural Network
CDH	Computational Diffie-Hellman
CNN	Convolutional Neural Network
CSTE	Split learning over Teacher Ensembles
DAE	Denoising Autoencoder
DBN	Deep Belief Network
DDH	Decisional Diffie-Hellman
DL	Deep Learning
DNA	Deoxyribonucleic acid
DNN	Deep Neural Network
DP	Differential Privacy
DRE	Direct-recording electronic
DSS	Digital signature standard
ECC	Elliptic Curve Cryptography
FL	Federated Learning
GAN	Generative Adversarial Network
GAP	Generative Adversarial Privacy
GC	Garbled Circuit
HE	Homomorphic Encryption
IID	Independent and Identically Distributed
LDP	Local Differential Privacy

LSTM	Long Short-Term Memory
LWE	Learning With Errors
ML	Machine Learning
MLP	Multilayer Perceptron
NLP	Natural Language Processing
non-IID	non-Independent and Identically Distributed
OT	Oblivious Transfer Protocols
PATE	Private Aggregation of Teacher Ensemble
PoI	Privacy of Input data
PoM	Privacy of Models
PoR	Privacy of output Results
PPDL	Privacy-Preserving Deep Learning
PPDM	Privacy-Preserving Data Mining
PPFL	Privacy-Preserving Federated Learning
PPML	Privacy-preserving machine learning
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMC	Secure Multi-party Momputation
SMS	Secure multi-party sum
SSC	Secure sum computation
SVM	Support Vector Machine
TEE	Trusted Execution Environment
TF-IDF	Term frequency – inverse document frequency
ZKP	Zero knowledge proof

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

1. Motivation

Deep learning (DL) has rapidly become a game-changer in the realm of machine learning, driving advancements across a wide range of applications, including image and speech recognition, natural language processing, and bioinformatics [1]. Its remarkable performance is intrinsically linked to the vast amount of data available for training. However, this dependence on large datasets raises significant privacy concerns, especially when dealing with sensitive information like personal health records, financial transactions, and confidential communications. In response to these challenges, the field of Privacy-Preserving Deep Learning (PPDL) has emerged to ensure that such data can be utilized without compromising privacy [2].

To address this challenge, Google Brain pioneered federated learning (FL), a groundbreaking technique that enables model training without requiring raw data to leave individual devices. Instead, FL facilitates the collaborative construction of a shared model by aggregating locally computed updates, ensuring that users' private data remains protected throughout the training process [3].

FL utilizes parameter averaging [4], which while preventing direct data leakage, can lead to unintentional information disclosure and compromise participant privacy [5]. Recent research focuses on mitigating this by incorporating differential privacy (adding noise) or cryptographic methods (secure multi-party computation) to enhance security during parameter sharing. However, despite these advancements, there's still room for improvement in security and efficiency when sharing local models [6].

Differential privacy (DP) techniques often require a trade-off between accuracy and privacy [7]. While they provide a layer of privacy protection, they are not immune to vulnerabilities and can still be susceptible to certain types of attacks, such as membership inference or model inversion, which may compromise the privacy of individuals. On the other hand, cryptographic methods, especially Secure Multi-

Party Computation (SMC) protocols, offer a promising avenue for achieving both privacy and efficiency [8]. However, several challenges remain that hinder the practical application of these cryptographic approaches. Key limitations include the need to safeguard against collusion among participating parties, which remains a significant hurdle. Additionally, converting real numbers to integers for SMC computations can result in accuracy degradation, along with increased communication overhead and higher computational costs during the training process. Moreover, many advanced protocols require the continuous involvement of all parties, which can be impractical in complex, real-world scenarios involving heterogeneous and geographically distributed networks.

This thesis addresses these crucial limitations by proposing and enhancing SMC protocols to enable secure and efficient federated learning in practical scenarios.

2. Thesis Objectives

Driven by the pressing need highlighted in this thesis, the dissertation focuses on enhancing the security of distributed deep learning model training by integrating FL with SMC techniques. Specifically, it tackles the challenge of protecting the privacy of parameter sharing among local models, particularly in scenarios where participants are semi-trusted and the possibility of collusion exists. This issue is especially critical in contexts involving a large number of participants, where current solutions have proven inadequate.

To achieve this goal, the research focuses on two key objectives:

- ***Developing efficient secure sum protocols for real number vectors:*** This objective involves designing innovative SMC protocols that are specifically tailored for handling real number vectors in a distributed setting. These protocols are developed to ensure security even when participants are semi-honest and may collude.
- ***Building robust distributed deep learning training frameworks:*** The research also explores the integration of these proposed SMC protocols with FL. This integration aims to construct secure training frameworks for distributed deep

learning models, applicable in both centralized and decentralized network environments where participants may not fully trust one another and collusion is a potential risk.

3. Research Subject and Scope

This dissertation centers on safeguarding privacy in the training of distributed deep neural networks through federated learning and SMC protocols, leveraging cryptographic techniques.

The selection of this research topic is motivated by the following considerations:

- Federated Learning: This method is highly effective for training distributed deep learning models, offering accuracy comparable to centralized training without privacy guarantees. It also minimizes the risk of data leakage by not sharing raw data [6].
- SMC Protocols based on Cryptographic Tools: These protocols can ensure the model's accuracy without compromising it, unlike other techniques such as data augmentation or perturbation [9].

4. Research content

With the research objective outlined above, the thesis identifies four main research contents, presented in the following chapters of the dissertation:

- Conduct a thorough review of the PPDL challenge, examining the strengths and weaknesses of various methodologies. The focus will be on FL, delving into its potential as a promising solution for ensuring privacy in deep learning systems.
- Develop efficient secure multiparty summation protocols for real-number vectors in semi-trusted environments, considering the possibility of collusion among participants. The research will provide a thorough analysis of the performance and security features of the proposed protocols.
- Develop distributed deep learning training protocols for both centralized and

decentralized networks using the proposed SMC protocols.

- Experiment and evaluate the proposed distributed deep learning training protocols using various datasets and deep learning architectures. Compare the experimental results with existing approaches to highlight the contributions of the dissertation.

5. Research Methodology

The dissertation's research methodology follows a well-defined roadmap to achieve its objectives. The first step involves an exhaustive survey of relevant domestic and international scientific literature on PPDL. This includes a thorough exploration of online databases like ACM Digital Library, IEEE Xplore, and Google Scholar, as well as reports from esteemed scientific conferences such as USENIX, Blackhat, SOICT, ICCM, and FAIR. These sources serve as the primary foundation for the theoretical research, providing crucial knowledge and insights into the PPDL domain.

Following the literature review, a systematic analysis identifies existing challenges in PPDL that require further investigation and resolution to align with the research goals. This process involves meticulously examining the literature to pinpoint gaps and limitations in current distributed deep learning model training approaches.

Next, the dissertation rigorously assesses the security and efficiency of the proposed SMC protocols using mathematical theory. This evaluation encompasses the protocols' security along with a detailed analysis of their communication and computation costs.

To validate the proposed distributed deep learning training protocols, experimental evaluations are conducted on various datasets and network architectures. The efficacy of the protocols is meticulously assessed by analyzing performance metrics like accuracy, convergence rate, and training time.

A comparative analysis is then performed against existing approaches to highlight the unique contributions of the dissertation. Finally, the obtained results are thoroughly analyzed and compared with related research works. The dissertation emphasizes its contributions in terms of scientific novelty, practical significance, and potential impact, solidifying its position within the broader landscape of PPDL research and innovation.

6. Main contributions

The dissertation focuses on addressing the research contents mentioned above. The contributions can be listed as the results of this dissertation, specifically including:

Contribution 1. This thesis introduces *three innovative SMC* protocols specifically designed for secure summation of real-number vectors. These protocols ensure robust security without depending on a trusted third party or necessitating the involvement of more than two participants who remain non-colluding. This is accomplished through the use of a homomorphic encryption scheme, renowned for its robust semantic security. The following is a detailed overview of the three proposed protocols:

- *Integer quantization combine with modified homomorphic encryption protocol:* This method utilizes an integer quantization technique coupled with a modified ElGamal cryptosystem. Real numbers are compressed with different precision levels, achieving differential privacy through inherent noise and computational security through cryptography. However, careful selection of compression ratios is crucial to balance model accuracy with communication and processing costs associated with large integer computations.
- *Mask matrix combined with homomorphic encryption protocol:* Addressing the limitations of the first protocol, this approach utilizes a mask matrix technique with a modified Elliptic Curve Cryptography (ECC) system. This combination aims to achieve high accuracy while minimizing communication and processing overhead.
- *Mask matrix combined with homomorphic encryption protocol and authentication sub-protocol:* While the first two protocols offer efficiency and accuracy, they lack data authentication and are vulnerable to membership spoofing at-

tacks. To address this, the thesis proposes a third protocol that combines a random noise masking matrix with the ElGamal cryptosystem, hashing functions, and digital signatures. This composite approach ensures data secrecy, privacy, and participant validity for robust secure vector summation.

The three secure vector summation protocols introduced in this work, each employing a distinct method, allow a group of n participants to compute the sum of their private inputs securely, within a semi-honest framework. These protocols are designed to withstand collusion among up to $n-2$ parties and are capable of handling real-number values, making them highly suitable for federated learning applications.

Contribution 2. Building upon the proposed SMC protocols, this thesis makes a second key contribution: *two novel deep learning training frameworks*. These frameworks leverage the SMC protocols to evaluate their effectiveness in training federated learning models across both centralized and decentralized network configurations. This evaluation encompasses both theoretical and empirical aspects. Theoretically, the research analyzes the privacy guarantees and communication overhead associated with the training process using these frameworks. Empirically, extensive evaluations are conducted on diverse datasets, including the balanced MNIST dataset (handwritten digits), the imbalanced SMS Spam dataset (text messages), and the CSIC 2010 dataset (cybersecurity). A range of deep learning architectures, including Convolutional Neural Networks (CNNs), Character-level CNNs (CLCNNs), and Long Short-Term Memory (LSTM) networks, were utilized in these experiments. The outcomes strongly support the effectiveness of the proposed methodology in achieving high accuracy. For example, the model attained a baseline accuracy of 97% on the MNIST dataset after just 10 training iterations, and 50 iterations for the SMS Spam dataset. Moreover, the approach proves to be resilient in heterogeneous distributed networks, even when data distributions are non-identical and imbalanced. Notably, the empirical results show a fivefold reduction in the number of training iterations needed to reach baseline accuracy, compared to the Downpour SGD algorithm, highlighting the substantial efficiency improvements of the proposed frameworks. This thorough evaluation underscores the potential of the proposed SMC protocols for enabling privacy-preserving federated learning across a variety of network setups and data types.

Our approach provides a distinct advantage over differential privacy techniques by ensuring strong cryptographic-level privacy while maintaining superior model utility. This results in enhanced efficiency and practical applicability, making the protocols highly suitable for real-world federated learning scenarios.

The proposed protocols are innovative and tailored specifically to handle floating-point real numbers. They focus on a critical security goal: safeguarding local models from honest-but-curious participants who may try to access sensitive data. Additionally, these protocols are designed to be resilient, ensuring protection even if a subset of participants, up to $n-2$ out of n, colludes within the semi-honest framework of privacy-preserving deep learning.

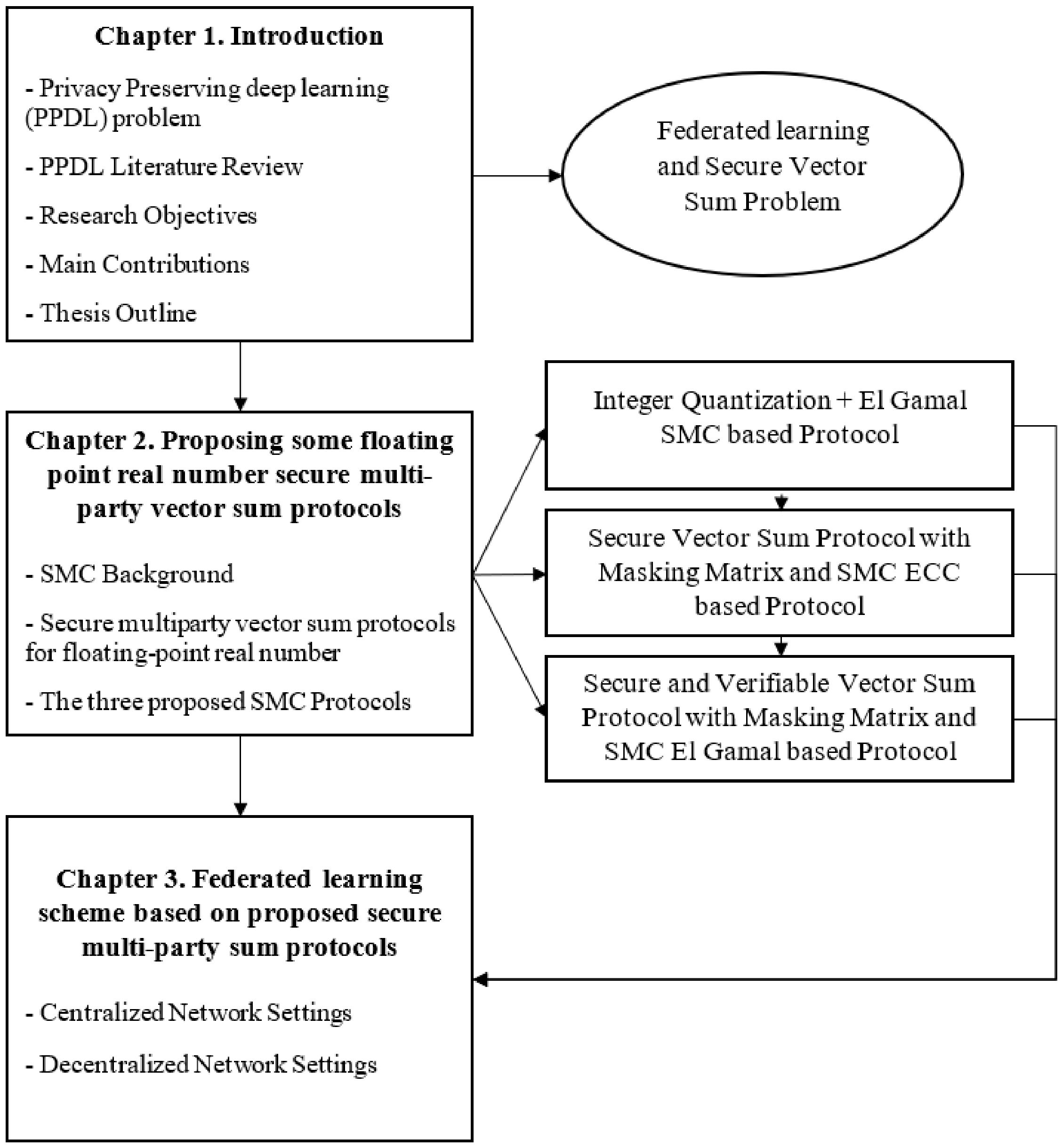
7. Organization of the dissertation

Based on the research findings, this dissertation is organized as follows: an introduction, three main chapters, and a conclusion outlining potential future directions for the topic. The detailed structure of the dissertation is presented below:

Introduction: The dissertation highlights the urgency and scientific relevance of the research topic, which forms the basis for defining the research objectives, scope, content, and methods employed in the study.

Chapter 1: This chapter explores the field of PPDL, critically assessing the advantages and drawbacks of three primary approaches: input sharing, model sharing, and output sharing. After a comprehensive evaluation, the dissertation focuses on addressing the challenges associated with training distributed deep learning models through a model-sharing approach. In this method, local model parameters are shared to build a more accurate global model, all while protecting the privacy of the local training data. However, sharing model parameters introduces potential risks, including attacks like membership inference and model inversion, which can inadvertently lead to data leakage.

Chapter 2 introduces three innovative secure vector summation protocols: (1)


Chapter 1. Introduction
- Privacy Preserving deep learning (PPDL) problem
- PPDL Literature Review
- Research Objectives
- Main Contributions
- Thesis Outline
Federated learning and Secure Vector Sum Problem
Chapter 2. Proposing some floating point real number secure multi-party vector sum protocols
- SMC Background
- Secure multiparty vector sum protocols for floating-point real number
- The three proposed SMC Protocols
Integer Quantization + El Gamal SMC based Protocol
Secure Vector Sum Protocol with Masking Matrix and SMC ECC based Protocol
Secure and Verifiable Vector Sum Protocol with Masking Matrix and SMC El Gamal based Protocol
Chapter 3. Federated learning scheme based on proposed secure multi-party sum protocols
- Centralized Network Settings
- Decentralized Network Settings

Integer quantization combined with a modified homomorphic encryption protocol, (2) Mask matrix combined with a homomorphic encryption protocol, and (3) Mask matrix combined with a homomorphic encryption protocol along with an authentication sub-protocol. In this chapter, the proposed protocols will be evaluated, and their security as well as execution cost will be analyzed and demonstrated.

Chapter 3 investigates the application of the proposed SMC protocols in federated learning. It presents two novel deep learning training protocols, designed to operate in both centralized and decentralized network environments. These protocols incorporate the SMC protocols to evaluate their effectiveness in training federated learning models. The evaluation spans a diverse set of datasets, including MNIST, SMS Spam, and CSIC2010, across three distinct network architectures: CNN, LSTM, and CLCNN, respectively.

Finally, the dissertation concludes, along with future research directions.

CHAPTER 1. PRIVACY PRESERVING DEEP LEARNING

This chapter provides a thorough exploration of PPDL, examining its foundational concepts, key challenges, and three main approaches: input sharing, model sharing, and output sharing. It critically assesses the strengths and limitations of each approach, ultimately concluding that model sharing—especially through Federated Learning—emerges as the most promising method for training distributed deep learning networks. Building on this insight, the dissertation aims to develop and propose appropriate SMC protocols to enhance the security of federated learning systems. The detailed analysis is outlined in **Publication 1**.

1.1. Deep learning

Observation vectors, or attribute collections, are crucial for training machine learning models, as their representation directly impacts performance. For example, spam detection considers sender info, link trust, and attachments, while text is represented through word frequency vectors. Feature selection has traditionally been manual, time-consuming, and costly, often leading to oversimplified or non-generalizable attributes across different datasets or tasks, such as distinguishing cats from dogs versus cars [10].

Deep learning (DL), inspired by the brain's structure, uses multi-layered artificial neural networks for data processing. These networks consist of neurons that extract increasingly complex patterns from raw data, which form the foundation for model learning and accurate predictions [11]. CNN, as shown in Figure 1.1, use convolutional layers to specialize in feature extraction [12].

Deep learning involves two main phases: training and inference [11]. During training, the model starts with random weights and adjusts them using input batches to optimize parameters like weights and coefficients for the dataset. This optimization relies on the loss function $\mathscr{L}(W)$, which compares predicted values to actual labels, guiding the model to reduce errors. The loss function quantifies prediction discrepancies across the training set $\{x_1, x_2, \ldots, x_m\}$, refining the model's parameters

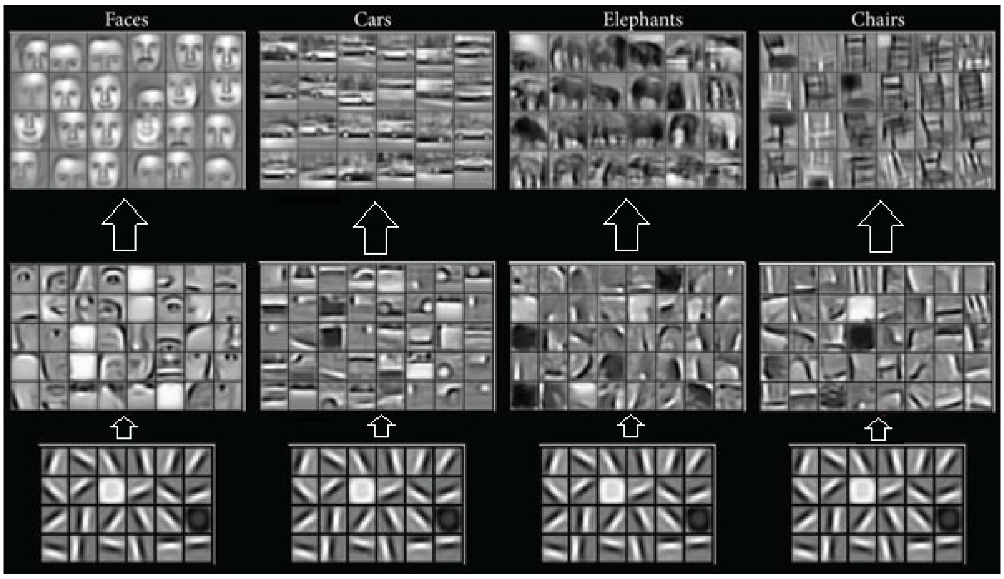


Figure 1.1: The process of deriving high-level features utilizing deep learning methods.

W through optimization. The loss function is mathematically represented in Equation 1.1.1.

$$\mathscr{L}(W) = \frac{1}{m}\sum_{i}\mathscr{L}(W, x_i). \tag{1.1.1}$$

DL models consist of multiple layers that perform complex, non-linear transformations on data, making it challenging to find the optimal configuration for performance. The interplay between the model's architecture and activation functions complicates this process, resulting in a non-linear loss function that drives the training.

Finding the global minimum in this non-linear space is difficult, so optimization algorithms rely on iterative updates to parameters, aiming for a ”good enough” solution (local minimum). Techniques like Nesterov, Adagrad, and ADAM, derived from stochastic gradient descent (SGD), perform these updates to minimize the loss function [13].

SGD processes each data point individually, but frequent updates can be computationally expensive. Batch gradient descent, on the other hand, processes the entire dataset at once, which can be slow or impractical for large datasets. To strike a balance, minibatch gradient descent is commonly used. This approach updates the model's parameters using a randomly selected subset of data, denoted as B, in each step, improving both efficiency and stability by mitigating the influence of outliers [14].

$$g_B = \frac{1}{|B|} \sum_{x \in B} \nabla_W \mathscr{L}(W, x). \tag{1.1.2}$$

During optimization, the model's parameters, denoted by W, are iteratively adjusted to minimize the loss function. Each update moves the parameters in the opposite direction of the gradient, which indicates the direction and magnitude of the change needed to reduce the loss. This process continues until a satisfactory minimum is achieved:

$$w_j = w_j - \eta \frac{\partial \mathscr{L}_i}{\partial w_j}. \tag{1.1.3}$$

The learning rate (η) controls the size of the adjustments made to the model's parameters. Each full pass through the training data is called an epoch. The loss $\mathscr{L}_i$, calculated for the i-th mini-batch, is essential in guiding these adjustments. Minimizing this loss is the core goal of training [15]. The process continues until the model converges to a stable state, typically reaching a local minimum. Different gradient descent optimization algorithms may vary in learning rates and incorporate additional techniques to enhance efficiency.

Once trained, the model enters the prediction phase, where it takes unseen data as input to generate predictions. These predictions can be applied to a variety of tasks, such as image classification, speech recognition, and natural language processing.

Both training and prediction involve passing data through the model's layers (forward pass). However, during prediction, there is no back-propagation, as no weight updates are needed—only the generation of predictions.

Deep learning encompasses a wide range of architectures, each suited for different tasks. Multi-Layer Perceptrons (MLPs) are foundational models that include hidden layers to learn complex patterns. CNNs, inspired by the visual processing mechanisms in the brain, are primarily used for image recognition and computer vision tasks. They use convolutional layers to apply filters to input data, capturing spatial hierarchies and patterns at different levels of abstraction. Deep Autoencoders (DAEs) are designed for unsupervised learning, focusing on reconstructing input data to achieve efficient data encoding. Recurrent Neural Networks (RNNs), especially LSTM, excel at processing sequential data, such as text, by incorporating information from previous time steps. Generative Adversarial Networks (GANs) consist of two competing models: a generator that creates synthetic data and a discriminator that differentiates between real and generated data, driving the generator to improve its outputs. These diverse architectures demonstrate the power and flexibility of deep learning for solving a wide array of problems [16].

1.2. Privacy Preserving Deep learning

1.2.1. Privacy threats with DL models

Deep learning's power hinges on vast amounts of data, but this raises critical data security concerns. Figure 1.2 illustrates the key areas where data protection is paramount to protect sensitive information throughout the entire lifecycle (training, inference, and model sharing).

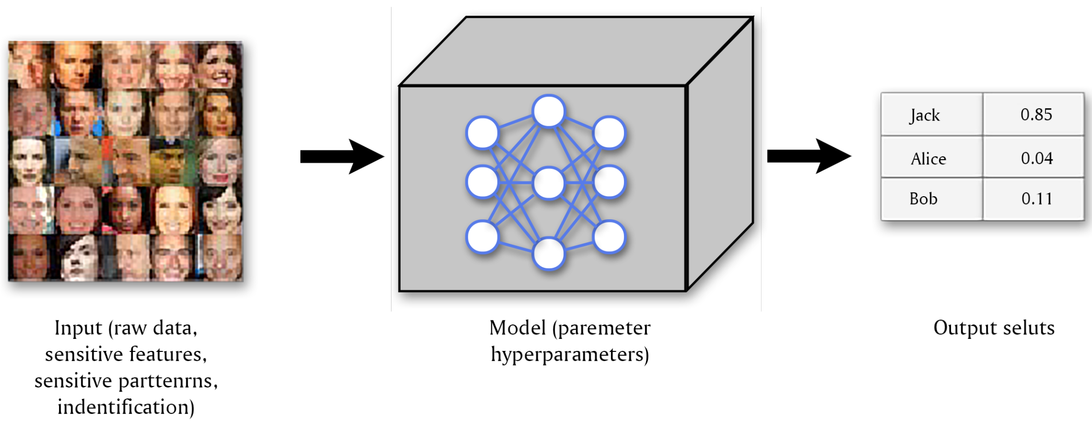

Jack 0.85
Alice 0.04
Bob 0.11
Input (raw data, sensitive features, sensitive parttenrns, indentification)
Model (paremeter hyperparameters)
Output seluts

Figure 1.2: Different data types in DL Models.

1.2.1.1. Privacy threats to input data

Input data can be leaked during collection or training, compromising confidentiality. Data leakage can occur in two primary ways:

- **Direct Leakage:** This happens when attackers gain unfettered access to the sharing data itself. This could be due to unencrypted transmission, compromised client devices, or even intercepted data during transfer. Additionally, there is a risk involving third-party servers, where intermediaries might intentionally access and harvest information, further exacerbating security vulnerabilities.
- **Indirect Leakage:** This type of attack is more nuanced, as the attackers don't directly access the data itself. Instead, they extract sensitive information by analyzing the trained model. They can do so by leveraging the model's predictions (referred to as a *black-box attack*) or by gaining insight into the model's internal parameters (known as a *white-box attack*).

Indirect data leakage attacks represent serious threats to the security of machine learning systems. Two common forms of such attacks are inversion attacks and inference attacks.

An *inversion attack* aims to reconstruct input data from a trained model by either predicting the model's outputs for specific inputs or using optimization algorithms to match inputs to desired outputs, exposing hidden sensitive information. Such attacks demonstrate the potential to uncover sensitive data indirectly by identifying patterns or unique traits in the training data through trained models or their public features [17]. Fredrikson et al.'s study showed how attackers could retrieve patient gene information using model predictions on drug dosage and variables like height, age, and weight, requiring only access to the model's prediction API or its outputs [18]. Deep learning models, due to their ability to retain information, compromise the anonymity of their training data [19]. This vulnerability allows adversaries to extract information from the training dataset using model-derived observations, as evidenced by the reconstruction of facial images from original photographs (Figure

1.3 [17]), assuming knowledge of the user's identity and the model's facial recognition capabilities, including the reliability of its predictions.

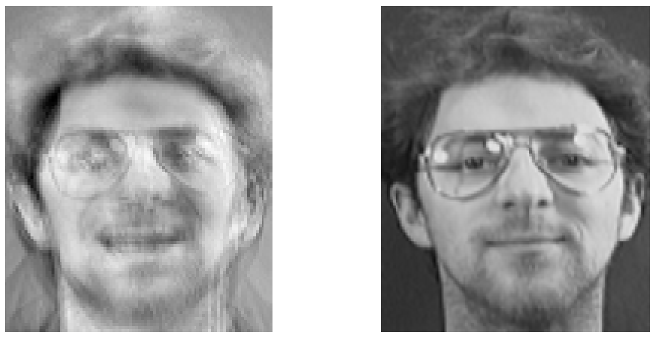


Figure 1.3: The left image shows the outcome of the inversion attack, while the right image represents the original version

Salem et al. [20] and Zanella-Béguelin et al. [21] demonstrated the vulnerability of systems to inversion attacks, particularly in online learning and language model updates using GANs. He et al. [22] further highlighted how these attacks can jeopardize query privacy in distributed learning environments, such as split learning, where model components are distributed across participants. In this setup, attackers can reconstruct part of the input data without needing full access to the dataset or computational resources. Hitaj et al. [23] showed that even with decentralized training and noise addition, GANs can still be used to breach the system and retrieve original data, emphasizing the model's susceptibility to inversion attacks.

In contrast, an *inference attack* takes advantage of the information embedded in a trained model to infer details about the input data. Attackers use statistical analysis or machine learning techniques to train secondary models based on the output of the target model, allowing them to uncover insights about the training data. Attribute inference attacks, for example, enable attackers to deduce specific characteristics of data records. Yeom et al. [24] studied this type of attack, where attackers assess the model's performance loss for different hypothetical values of a sensitive attribute. By comparing these losses to the actual data, attackers can determine the most likely value for that attribute.

Membership inference attacks aim to determine whether a specific data point was used in training a model. Shokri et al. [25] explored these attacks by having an adversary query the model to obtain its prediction confidence score, which reveals whether a data record was part of the training set. They created an attack model by generating new data through techniques like model inversion and noise addition, then used it to detect if inputs were in the original training set, based on the model's confidence in familiar records. Their method, tested on a multi-layer model with data from Google retail transactions and a Texas hospital, achieved accuracy rates of 94% and 70%, respectively. This demonstrates how attackers can extract sensitive information from models, even with limited access.

Truex et al. [26] proposed a framework to understand membership inference attacks, highlighting the risk of deep neural networks memorizing training data and becoming vulnerable. Recent work by Salem et al. [27] and Song et al. [28] shows that even low-resource attacks can be effective. Additionally, Hayes et al. [29] explored how these attacks extend to models using GANs, broadening the scope of their potential impact.

1.2.1.2. Privacy threats to trained models

The training model is considered a highly valuable and confidential asset by its owner, making it a prime target for attackers seeking to replicate or steal it. One common attack method is the *model stealing attack*, where attackers attempt to reconstruct the model's parameters by analyzing its prediction outputs, especially the confidence scores, for a given set of inputs. Tramer et al. [30] demonstrated how attackers could infer a deep learning model's parameters by studying the correlation between inputs and their corresponding outputs. However, such attacks are less effective when the attacker only has access to final predictions without any confidence score information. In addition to targeting model parameters, attackers may also gain insights by exploiting other aspects of the model, such as its hyperparameters [31], architecture [32,33], decision boundaries [34–36], or by simulating its functions [37,38], as shown in multiple studies.

1.2.1.3. Privacy threats in prediction outcomes

In many cases, prediction outputs are considered sensitive information [39]. For example, a service provider that obtains a user's identity through registration might offer predictions on topics such as health diagnoses or financial forecasts, thereby gaining access to a wealth of personal data. Moreover, by storing the results of these predictions, the provider can infer additional private details. For instance, if predicting liver disease with high confidence using alcohol addiction as a factor, the diagnosis of liver disease could indirectly suggest a higher likelihood of alcohol dependency for that user.

1.2.2. Overview of privacy challenges in Deep Learning

The primary objective of PPDL is to protect both the confidentiality of training data, trained models and the prediction outcomes. There are four key privacy challenges in PPDL: 1) Data publishing, where an entity shares data with others for model training due to limited resources; 2) Data collection, which focuses on acquiring data from diverse sources without infringing on privacy; 3) Prediction service, ensuring that both the input data and the model's predictions remain confidential in prediction services; and 4) Distributed Learning, which aims to aggregate data from multiple sources to improve model performance without compromising privacy.

1.2.2.1. Privacy-preserving data publishing

In the realm of data publishing (Figure 1.4), an entity $\mathscr{D}$ with dataset D may lack sufficient computational power to train deep learning models, prompting the practice of outsourcing data for model training and development, commonly referred to as data outsourcing [40]. This practice has become widespread in the digital era, as organizations often share data with external experts for analysis or make it publicly accessible for research and competitions [41]. A notable example is Netflix's release of anonymized user data for the Netflix Prize, aimed at improving recommendation algorithms [42]. However, this raises significant privacy concerns, as there is a risk of data leakage and re-identification [43–45]. Narayanan et al. demonstrated the

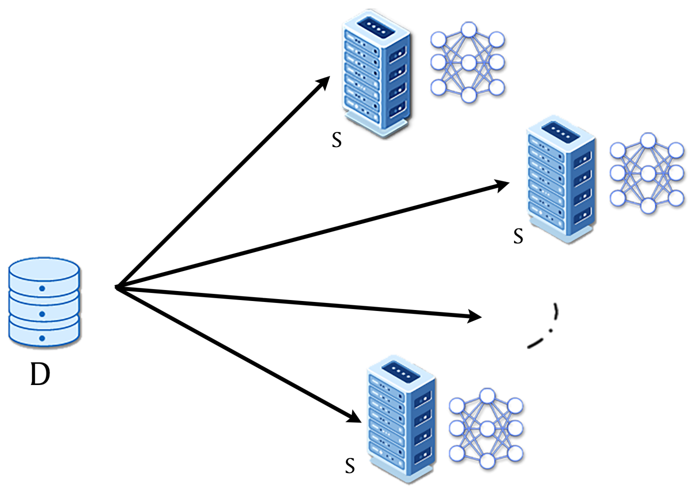


Figure 1.4: Data Publishing

feasibility of re-identifying individuals, exposing the limitations and potential vulnerabilities of current anonymization techniques [46, 47].

1.2.2.2. Privacy-preserving user data collection

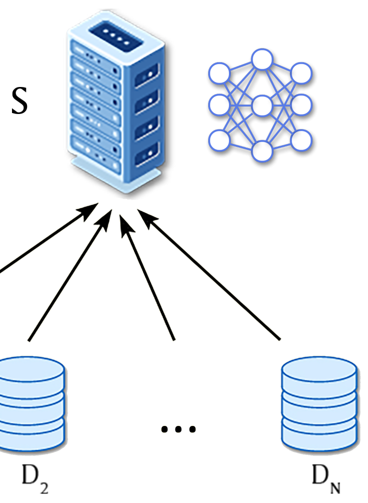


Figure 1.5: Training data collection from many sources

In this scenario, an entity responsible for model training gathers data from multiple sources (Figure 1.5), each of which seeks to safeguard its own data privacy [48–50]. Despite privacy concerns, these sources upload their data to a centralized server for model training. This process can be classified as horizontal distribution when each source provides data with identical features, or vertical distribution when different sources contribute distinct types of data. Unlike data publishing models, where multiple entities may share both data and computational resources, only one entity in this case assumes the role of the training server. Data contributors, however, do not participate in the computational processes of model training. Privacy risks, particularly in the form of white-box attacks targeting the training server, could expose sensitive information from participants. To address these concerns, solutions such as model encryption or secure sharing frameworks like SecureNN [51] offer promising ways to mitigate potential vulnerabilities.

1.2.2.3. Privacy-preserving prediction service

In this framework, a server (S) and a user (U) interact, where S utilizes a pre-trained deep learning model—developed from a dataset—to offer prediction services (Figure 1.6). The user (U) requests a prediction from the server, which responds with the output. The main challenge is to protect sensitive data transmitted by the user for prediction purposes, as well as the confidentiality of the prediction results. Additionally, the proprietary nature of the model and the dataset used for training must be safeguarded against unauthorized access. This requires securing several elements: the user's input data, the resulting predictions, the model itself, and the underlying training data. Effectively addressing these privacy concerns is essential to maintain the confidentiality of all parties involved—user data, model outputs, and intellectual property.

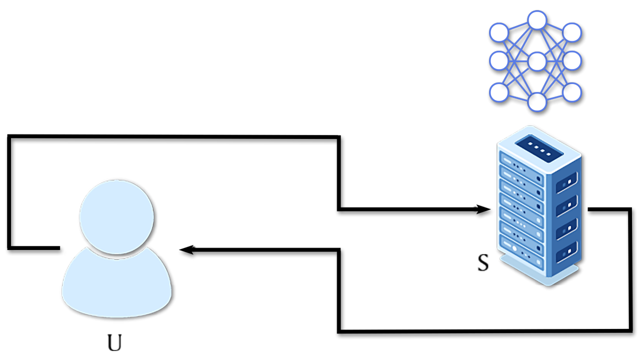
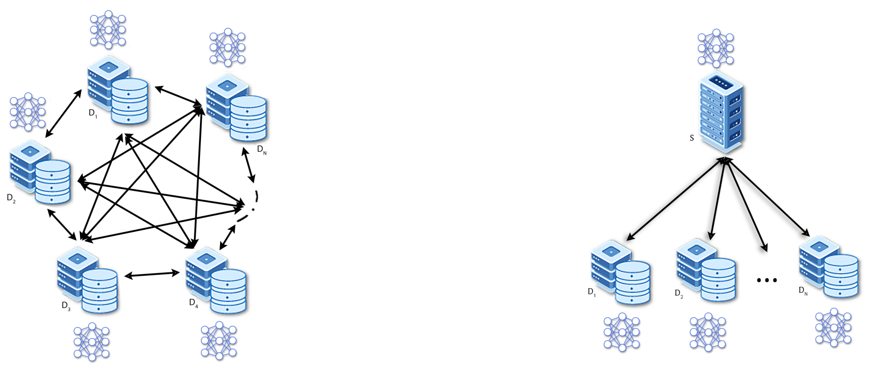


Figure 1.6: Privacy Guarantee for prediction service

1.2.2.4. Privacy-preserving distributed training

This approach aims to develop training protocols that allow participants to contribute to model training without exposing their private data. Each participant processes a portion of the training on their own dataset and subsequently shares the results to collectively build a global model, thus safeguarding data privacy [52]. The method distinguishes between centralized networks, where local models are sent to a central server for aggregation, and decentralized networks, which rely on direct peer-to-peer communication [53]. The architecture shown in Figure 1.7 illustrates these two configurations.



(a) Decentralized network settings (b) Centralized network settings

Figure 1.7: Distributed deep learning model

Data distribution plays a critical role in devising privacy solutions, generally falling into horizontally or vertically distributed models [54]. In the horizontal model, participants have data with common attributes but differing records, useful in systems like anomaly detection in network traffic. The vertical model involves participants holding the same records but different attributes, seen in collaborations among banks, tax authorities, and hospitals [55]. An example given by Jaideep Vaidya et al. shows the combination of medical data with telephone usage to study health impacts [56].

In both horizontal and vertical distributions, successful training requires data exchange among parties, posing privacy concerns. The primary challenge is developing methods that maintain training data privacy while leveraging necessary information for model development.

This dissertation delves into the intricate challenge of developing privacy-preserving distributed deep learning models, specifically tailored for scenarios involving horizontally partitioned data.

1.3. Privacy Preserving Primitives

This section delves into a selection of established data privacy strategies highlighted in academic literature, focusing on their applied efficacy. Within the domain of PPDL, a variety of approaches draw on these fundamental strategies. We categorize these primitives into three main groups. The first, anonymization, involves modifying quasi-identifiers and eliminating direct identifiers to safeguard individual privacy. The second, cryptographic methods, leverage well-established protocols to ensure data confidentiality. Lastly, data obfuscation employs the strategic insertion of noise to veil the original data, maintaining its utility for machine learning applications while protecting sensitive information.

1.3.1. Anonymization

To protect privacy during model training, data is detached from its owner's identity, keeping the data unchanged. Simple anonymization, like removing names or addresses, is often inadequate, as shown by the Netflix Prize case, where researchers identified users from anonymized movie ratings using additional data from the Internet Movie Database [46]. K-anonymity aims to make data re-identification difficult by ensuring each data point is indistinguishable from at least k-1 others [57], but it struggles with high-dimensional data, leading to advanced concepts like l-diversity [58] and t-closeness [59], which are beyond this thesis's scope.

1.3.2. Cryptographic techniques and Secure Multiparty Computation

The concept of SMC first emerged in the late 1970s, with early contributions from A. Shamir, R. Rivest, and L. Adleman, who sought to enhance privacy in telephone-based poker games and mitigate fraud risks [60]. However, it was formalized in 1982 when Yao introduced the Garbled Circuit (GC) protocol [61], which enabled secure information exchange by representing computational functions using

Boolean logic circuits. In 1987, Goldreich et al. further expanded SMC's scope, enabling secure collaborative computation without the need for a trusted third party [62]. Since then, SMC has become crucial in various domains, including secure voting systems [63], online auctions [64], and data protection, particularly in privacy-preserving data mining (PPDM) and privacy-preserving deep learning (PPDL) applications [65]. Among the key techniques used in SMC are oblivious transfer (OT), homomorphic encryption (HE), and secret sharing (SS), which will be explored in greater detail in the following sections.

1.3.2.1. Basic concept

Secure Multiparty Computation (SMC) is a cryptographic framework that involves distributed computing protocols, allowing multiple participants to collaboratively compute the value of a function while maintaining the confidentiality of their private inputs. Various definitions of SMC have been presented in existing literature [66–69]. For the purpose of this thesis, a generalized model will be adopted, encompassing the most relevant privacy-preserving approaches applicable to deep learning models.

Definition 1.3.1. *Let n ($n \geq 2$) represent the number of participants in the distributed computing network. Each participant $i \in \{1,2,\dots,n\}$ holds an input value $x_i \in X_i$. The function f is then defined as a multi-party computation function, as given below:*

$$\begin{gathered} f : X \to Y \\ \bar{x} = (x_1, x_2, \dots, x_n) \mapsto f(\bar{x}) = (f_1(\bar{x}), f_2(\bar{x}), \dots, f_n(\bar{x})) \end{gathered} \tag{1.3.4}$$

where $X = \{\bar{x} : \bar{x} = (x_1,\dots,x_n)\}$ and $Y = \{y : y = (f_1(\bar{x}),\dots,f_n(\bar{x}))\}$ and X_i is value space for each x_i.

Every party i aims to obtain the specific component i of the function $f(x_1,\dots,x_n)$ while keeping their private input x_i. This is denoted by $f_i(x_1,\dots,x_n)$.

A multi-party computation function f can be classified into two types:

- Symmetric functions yield identical results for all participants involved. In this scenario, f_i and f_j are equal for all $i \neq j$.

- Asymmetric functions result in different outcomes for different participants. That is, $f_j(x_1,\ldots,x_n) \neq f_i(x_1,\ldots,x_n)$ when $j \neq i$ and $i, j \in \{1,\ldots,n\}$

In cryptography, SMC involves collaborative computation of a function f by entities while safeguarding their private input values against potential malicious behavior from adversaries. These protocols collectively constitute SMC protocols.

In contrast to traditional cryptographic domains like encryption or digital signatures, adversaries in SMC protocols can include participating members or external entities controlling internal members. Therefore, when evaluating an SMC protocol, three categories of entities emerge:

- **Honest parties** refers to members who strictly adhere to the rules outlined by the protocol.
- **Corrupted parties** are participants who either collude with others involved in the protocol or are manipulated by external entities to execute harmful actions against the honest parties.
- **External adversaries** are entities that exert control over certain members within the protocol, orchestrating actions that are harmful to the honest parties involved.

1.3.2.2. Threat models

As illustrated in Figure 1.8, we categorize adversarial attack assumptions in SMC based on the adversary's behavior, level of power, and the types of corruption they may introduce.

Adversarial behavior

Adversarial behaviors in SMC model are typically divided into two distinct types: **semi-honest** and **malicious**. The semi-honest, or honest but curious model, assumes that while all parties may adhere to the prescribed security protocols, those compromised might still seek to gain unauthorized access to sensitive information

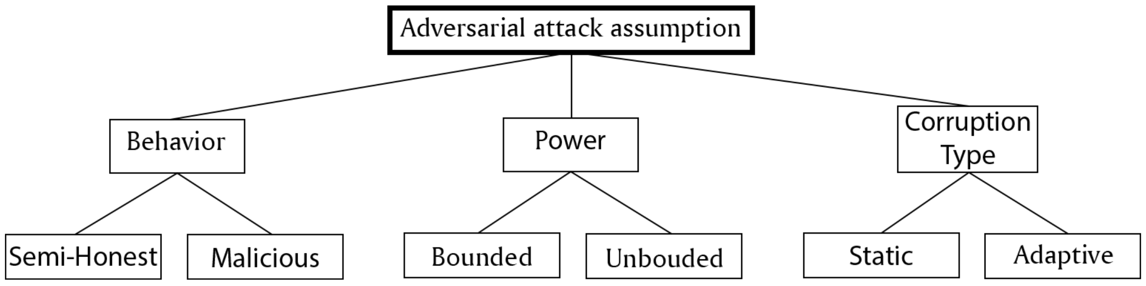


Figure 1.8: Adversarial attack assumptions in PPDL

through passive means, without actively harming the system or its users. This approach is commonly applied in SMC contexts.

On the other hand, the malicious model, also known as the active adversary model, characterizes corrupted participants who engage in direct attacks against the system, potentially violating the established security protocols to achieve their aims. When such parties are capable of discontinuing their attacks, this scenario is often referred to as a fail-stop model, indicating a specific subset of malicious behavior where attacks can be ceased.

This thesis primarily focuses on the semi-honest model, which is suitable when participants engage in SMC protocols with integrity and adhere to protocol rules. For instance, maintaining individual account confidentiality is paramount in collaborative financial data analysis among institutions. The semi-honest model prevents curious parties from accessing others' private data through observations. Despite its relatively relaxed security measures, this model is a fundamental initial step toward enhancing overall security. This model plays a crucial role in designing protocols for the malicious model and facilitates the transition of secure protocols from the semi-honest to the malicious model [62]. Participants strictly adhere to computational rules in the semi-honest model, making non-collusion assumptions unreasonable [70]. Therefore, the thesis focuses on SMC using the semi-honest model, allowing collaboration among participants while permitting up to $(n-2)$ corrupted parties, with n representing the total number of participating data users in the protocol execution. In this scenario, it is assumed that the adversary possesses knowledge of corrupted parties' information and has access to communication channels, which could be authenticated or even public.

Adversarial power

Adversaries are categorized as computationally **unbounded** or computationally **bounded** based on the extent of their attack capabilities. In a computationally unbounded setting, adversaries possess infinite computational power, representing the ideal or perfect adversary. This scenario is mostly theoretical and impractical in real-world contexts, often applied in theoretical information security studies.

Conversely, a computationally bounded adversary operates within constrained computational capabilities. Cryptographic assumptions, often involving polynomial time, are necessary to model attacks within this framework. As previously stated, the adversary possesses control over $(n-2)$ corrupted internal parties without knowledge of the honest entities involved. The communication channels among these parties are either authenticated or public, granting the adversary the capability to intercept transmitted messages. Moreover, the adversary's computational capacity is constrained, operating within a (probabilistic) polynomial-time framework [70]. This limitation signifies their restricted ability to perform extensive computations or attacks. This model aligns with the realistic assumption about the adversary.

Adversarial corruption type

Adversaries in SMC are categorized into **static** or **adaptive** types, contingent upon their method of selecting targets for corruption. In a static adversary framework, the identity of corrupted participants is predetermined from the start. Once designated as honest or corrupt, these participants maintain their roles throughout the process.

Contrastingly, the adaptive adversary framework allows for the flexibility of choosing which participants to corrupt as events unfold, making decisions based on the evolving context of the situation. This dynamic approach means that an individual previously acting in good faith could be compromised mid-execution. Despite this, within the adaptive model, there exists the potential for a corrupted individual to be influenced or pressured back into acting in an honest manner, highlighting the fluidity and complexity of participant roles under this classification.

1.3.2.3. Security definition

This section covers essential preliminaries crucial for establishing the standard security definitions in the field of SMC.

- **Ideal/real simulation paradigm**

Goldwasser and Micali [71] introduced the notion of semantic security in public-key cryptography. They argued that if an adversary can derive any information about the plaintext from the ciphertext, this information should not exceed what could be learned without any input. The idea of ”no input” represents an ”ideal world” scenario [72]. Thus, a system is considered secure in the ”real world” if the adversary gains no more information from the ciphertext than it would in the absence of any input.

The comparison between the ”real world” and the ”ideal world” forms the basis of the ”ideal/real simulation paradigm,” which is a fundamental approach for evaluating security. This paradigm sets the highest standard for security models, particularly when dealing with adversaries exhibiting malicious behavior.

In the context of SMC, the ideal world assumes the existence of a trusted party that facilitates computations among participants without introducing any security concerns. In contrast, the real world operates without such trust, requiring participants to remain vigilant about potential threats from one another. The security of a protocol is determined by comparing the outcomes of its execution in the real world with those in the ideal world. A protocol is deemed secure if any information an adversary could feasibly obtain in the real world could just as feasibly be obtained in the ideal world [62].

- **Negligible function**

Let n denote the security parameter, often associated with the key length, which determines the complexity of problems such as discrete logarithms and large integer factorization, making them infeasible to solve in polynomial time. The definition of a negligible function, as outlined in [62], is presented below.

Definition 1.3.2. *[62] A function $\mu(u)$ is considered negligible with respect to n if, for any positive polynomial $p(.)$, there exists a non-negative integer N such that for all $n > N$:*

$$\mu(u) < \frac{1}{p(n)}$$

- **Computationally indistinguishable**

Definition 1.3.3. *[62] Let $X(n,a)$ and $Y(n,a)$ be two random ensembles indexed by (n,a). The distributions $X = \{X(n,a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ and $Y = \{Y(n,a)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ are said to be "computationally indistinguishable" (denoted as $X \overset{C}{\equiv} Y$) in polynomial time if, for every probabilistic polynomial-time algorithm D, there exists a negligible function $\mu(n)$ dependent on n, such that for all $a \in \{0,1\}^*$:*

$$|\Pr[D(X(n,a)) = 1] - \Pr[D(Y(n,a)) = 1]| \leq \mu(n).$$

This indicates that no efficient algorithm D can differentiate X from Y with a significant probability of success, as any deviation in outcomes is limited by a negligible function.

In SMC, these parameters are defined as follows:

- n: Represents the security parameter, which defines the level of cryptographic difficulty.
- a: Denotes the input provided to the SMC protocols.
- X: Refers to the output generated by the SMC protocols in an ideal-world scenario.
- Y: Corresponds to the output produced by the SMC protocols in a real-world environment.

- **Standard definition of security**

This section presents the formal definition of security for an SMC protocol operating within the semi-honest model, utilizing public communication channels. The definition is based on the foundational framework of secure multiparty computation outlined in [62].

Definition 1.3.4. *The protocol* Π *is said to privately compute the function* f *in the presence of* t *corrupted parties if, for every subset* $I \subseteq \{1,2,\ldots,n\}$ *with* $\|I\| = t$*, there exists a probabilistic polynomial-time algorithm* $\mathbb{M}$ *such that:*

$$\{\mathbb{M}(I, \bar{v}_I, f_I(\bar{v}), f(\bar{v}))\}_{\bar{v}\in(\{0,1\}^*)^n} \overset{C}{\equiv} \left\{VIEW^{\Pi}_{A,I}(\bar{v}), \mathrm{OUTPUT}^{\Pi}(\bar{v})\right\}_{\bar{v}\in(\{0,1\}^*)^n},$$

where:

- $VIEW^{\Pi}_{A,I}(\bar{v})$*: Represents the view of the* t *corrupted parties along with all messages observed by the adversary* A *during the execution of the protocol* Π *with input* $\bar{v} = (v_1, \ldots, v_n)$*.*
- $\mathrm{OUTPUT}^{\Pi}(\bar{v})$*: Denotes the output of all parties involved in the protocol* Π*. When* f *is deterministic,* $\mathrm{OUTPUT}^{\Pi}(\bar{v}) \equiv f(\bar{v})$*, satisfying:*

$$\{\mathbb{M}(I, \bar{v}_I, f_I(\bar{v}))\}_{\bar{v}\in(\{0,1\}^*)^n} \overset{C}{\equiv} \left\{VIEW^{\Pi}_{A,I}(\bar{v})\right\}_{\bar{v}\in(\{0,1\}^*)^n}.$$

- $\overset{C}{\equiv}$*: Denotes computational indistinguishability.*

In essence, the protocol ensures that the view of the corrupted parties, combined with the information an adversary can observe, is computationally indistinguishable from the output that could be simulated with access only to the inputs and function outputs. This guarantees the privacy of the computation.

Let's consider a scenario where four participants are involved in a protocol to compute a function f, specifically the sum of their respective values v_1, v_2, v_3, and v_4. Suppose that two of these participants, U_1 and U_2, are corrupted parties who are colluding with an attacker A. In this case, the information available to the attacker A, represented by $\left\{\mathrm{VIEW}^{\Pi}_{A,I}(\bar{v}), \mathrm{OUTPUT}^{\Pi}(\bar{v})\right\}_{\bar{v}\in(\{0,1\}^*)^n}$, includes:

- The values v_1 and v_2 obtained through collusion with U_1 and U_2,
- The public result of the sum $\mathrm{OUTPUT}^{\Pi}(\bar{v}) = v_1 + v_2 + v_3 + v_4$, which is shared among all participants.

Due to the protections offered by the protocol, the attacker A should only be able to obtain these three pieces of information (i.e., v_1, v_2, and the final sum). Additionally, A might have access to the encrypted forms of v_3 and v_4, but not their actual values.

In this setting, each participant shares a "masked" or encrypted version of their number. The protocol is designed to ensure that, even if A uses all the information received during the protocol execution, they cannot accurately determine the exact values of v_3 and v_4 held by U_3 and U_4. The only information that A can deduce is the final sum (output), not the individual contributions of v_3 and v_4.

The protocol Π is considered secure if the adversary A gains no additional knowledge from participating in the protocol beyond what can be inferred from the final output (e.g., the sum). This means that the adversary's view, VIEW, can be computationally simulated using only the result of the function f, ensuring no unintended information is exposed.

As a result, the protocol safeguards the privacy of the inputs v_3 and v_4, even in scenarios where A collaborates with U_1 and U_2 in an attempt to breach security.

Additionally, the composition theorem plays a central role in designing SMC protocols under the semi-honest model (see Theorem 1.3.1). This theorem provides a foundational framework for establishing protocol security by proving the security of each constituent sub-protocol [62].

Theorem 1.3.1. *Let g and f be functions such that g is privately reducible to f - meaning that g can be expressed in terms of f while maintaining privacy. If there exists a secure protocol for privately computing f, then there exists a secure protocol for privately computing g.*

Since g is privately reducible to f, there exists a privacy-preserving transformation that maps the inputs and outputs of g to those of f without leaking any additional information. Given that f can be securely computed using a protocol that satisfies privacy and correctness, this secure protocol can be used in conjunction with the private reduction to compute g securely. The composition theorem for secure multiparty computation ensures that combining these two steps maintains the privacy and correctness properties, thereby yielding a secure protocol for g.

To construct SMC protocols, we often rely on fundamental cryptographic primitives such as Oblivious Transfer Protocols, Homomorphic Encryption, and Secret

Sharing. These building blocks play a crucial role in ensuring the security, privacy, and correctness of SMC protocols.

1.3.2.4. Oblivious Transfer Protocols

OT protocols are considered a viable approach in constructing SMC protocols, as indicated by the work of [73]. In their elementary manifestation, these protocols provide a specific problem scenario whereby the sender possesses two distinct messages. At the same time, the receiver desires to obtain only one of them without disclosing the specific message picked from the sender. The protocol, commonly called the 1-out-of-2 definition, is important in developing more intricate protocols [74]. The protocol is depicted in Figure 1.9.

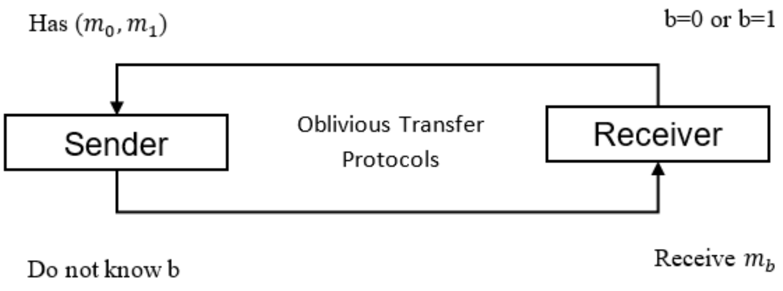


Figure 1.9: Oblivious Transfer Protocols

The protocol outlined in [75] extends its functionality to allow the receiver to request retrieval of a specific quantity, denoted as *k*, of messages from a pool of *n* messages stored by the sender. In this setup, it's ensured that (1) the sender remains unaware of the precise content the recipient will ultimately access, and (2) the recipient is limited to receiving a maximum of *k* units of the intended message. This protocol finds applications across various domains including sample and anonymous searches, database queries, and SMC [76].

1.3.2.5. Homomorphic Encryption

Homomorphic Encryption (HE) is a sophisticated cryptographic method enabling computations to be executed directly on encrypted data. The resulting encrypted outputs, when decrypted, align perfectly with the results of the same operations applied to the original plaintext [77]. This unique feature preserves the structure and integrity of the data while maintaining its confidentiality throughout the computation process.

For instance, in a homomorphic encryption scheme that supports addition, one can compute the sum of two encrypted messages, m_1 and m_2, by evaluating the expression $E(m_1 + m_2)$. Crucially, this calculation is performed using only the encrypted values, $E(m_1)$ and $E(m_2)$, without ever needing to decrypt them or access the original plaintexts. Here, the function E denotes the encryption operation within the homomorphic encryption scheme. The concept of homomorphic encryption is fundamentally tied to the idea of preserving the algebraic operations of plaintexts within the encrypted domain. To formalize this, we define a cryptographic system as homomorphic if it maintains a specific relationship between operations on the plaintexts and their corresponding ciphertexts. This relationship is captured in the following definition:

Definition 1.3.5. *[78] A cryptographic system is said to be homomorphic with respect to the operation '$\circ_2$' if it fulfills the following equation:*

$$E(m_1) \circ_1 E(m_2) = E(m_1 \circ_2 m_2), \qquad \forall m_1, m_2 \in M, \tag{1.3.5}$$

where E is the HE scheme and M is the plaintext space or set of all possible messages.

HE can be categorized in the literature into three distinct types: partially homomorphic [79], strong (or somewhat) [80], and fully homomorphic [81]. Partial homomorphic encryption (PHE) is a cryptographic scheme that maintains the property of preserving homomorphism in ciphertexts with a single type of operation, such as addition or multiplication, performed an unlimited number of times. The RSA cryptosystem exhibits the property of homomorphism with multiplication operations performed on encrypted data, enabling what is commonly referred to as multiplication homomorphic encryption [82]. As described by Paillier in [83], the Pallier cryptosystem exhibits homomorphic properties specifically with addition operations, thereby earning the designation of addition homomorphic encryption. The BGN cryptosystem, also known as the Boneh-Goh-Nissim cryptosystem, is widely recognized as one of the most renowned HE schemes to date [84]. BGN exhibits homomorphism properties to both addition and multiplication operations. However, it is important to note that the range of permissible operations is limited, hence classifying it as a somewhat homomorphic encryption scheme rather than a fully homomorphic one. Gentry conducted the initial investigation towards fully homomorphic encryption [81]. After this initial study, several investigations have been conducted on fully homomorphic encryption schemes rooted in lattice theory, specifically focusing on the Learning With Errors (LWE) problem [85–87]. The Ring-LWE problem has also been explored in this context [88, 89]. Furthermore, other cryptosystems have been developed on the greatest common divisor problem [90, 91]. The implementation cost of these cryptosystems is substantial, hence restricting their current utility to a considerable extent. However, the application of HE has been significantly enhanced with the introduction of HE-supported cryptographic tools such as HElib [92], FHEW [93], and HEEAN [94]. These tools have facilitated the expansion of HE's application in several domains, including enhancing security in cloud computing services [95–98] or PPML [99]. According to the cited source, the computational costs of HE associated with current deep learning models remain significant, making their practical implementation challenging.

1.3.2.6. Secret sharing scheme

The secret sharing scheme with a threshold of (t,n), where t and n are integers that satisfy $0 \leq t < n$, offers a mechanism for securely distributing a confidential data element s among a set of n data pieces denoted as $s_1, s_2, \ldots, s_n$. This distribution is intended to be shared with n participants so that no group of less than t untrusted members can reconstruct the original confidential information s. In contrast, any group of $t+1$ or more participants can successfully reconstruct the shared confidential information s [67].

In a theoretical scenario, it is crucial to highlight that possessing t sharing components does not ensure knowledge of the secret value s. As a result, obtaining such information does not automatically grant access to confidential data. In simpler terms, if an attacker has no prior knowledge before acquiring the t sharing, they will not gain any useful information. Furthermore, even if an adversary obstructs participant t', where $n - t' \geq t + 1$, retrieving the secret value with a minimum of $t + 1$ remaining participants is still feasible.

The concept of secret sharing schemes was initially introduced by Adi Shamir and Blakley as a solution to address the issue of a single point of failure [67]. Furthermore, these methods possess the capability to be employed in a multitude of issues and serve as a fundamental component in threshold cryptography and SMC [100, 101].

SMC protocols offer a robust level of security and guarantee a heightened level of privacy. Nevertheless, the advancement of the protocols needs to be improved by performance-related concerns. To effectively utilize SMC in PPDL, addressing the computational burden associated with nonlinear activation functions like Sigmoid or Softmax is imperative. The cost of these functions is prohibitively high.

1.3.3. Data obfuscation techniques

Data obfuscation techniques encompass manipulating or creating data instances obtained from the primary dataset, subsequently employed to train a comprehensive model. This category encompasses several tactics, including additive perturbation, multiplicative perturbation, generative obfuscation, and data synthesis.

1.3.3.1. Additive perturbation

Additive obfuscation is commonly linked to differential privacy (DP) in the academic literature [102, 103]. DP is a formal and quantifiable measure to assess the efficacy of safeguarding data privacy. This solution addresses the issue of sharing private data in situations where participants need to analyze the statistical characteristics of the data while maintaining confidentiality regarding specific records within the dataset. DP enables estimating the privacy level associated with the training data utilized by the technique employed [104, 105]. Developing a model or algorithm that aims to reduce the potential danger and extent of data privacy disclosure based on a specified threshold becomes feasible by employing DP. From a mathematical perspective, the concept of DP guarantees that the data analysis outcomes, when applied to DP, remain consistent regardless of the presence or absence of a specific individual or object inside the dataset.

Definition 1.3.6. *[104] A randomized mechanism $\mathscr{M} : \mathscr{D} \rightarrow \mathscr{R}$, where $\mathscr{D}$ is the domain of inputs and $\mathscr{R}$ is the range of outputs, satisfies (ε, δ)-differential privacy if, for all pairs of adjacent datasets $d, d' \in \mathscr{D}$ (datasets that differ in at most one data point) and for any subset of possible outputs $S \subseteq \mathscr{R}$, the following condition holds:*

$$\Pr[\mathscr{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathscr{M}(d') \in S] + \delta,$$

where Pr *represents the probability of an event. If this condition is satisfied with $\delta = 0$, then the mechanism $\mathscr{M}$ provides ε-differential privacy, which ensures a stricter form of privacy by bounding the likelihood ratio of outputs without any additional error term.*

In the definition above, the symbol ε represents the metric used to quantify privacy loss. A decrease in the value of $epsilon$ corresponds to a decrease in the extent to which privacy is violated. Nevertheless, this phenomenon results in a decrease in accuracy. The symbol δ represents the likelihood that an attacker can accurately guess data within the given range ε. In practical applications, DP is commonly implemented by augmenting the original dataset with a specific quantity of noise data. In general, random noise is generated using Gaussian or Laplacian distributions to maintain the statistical characteristics of the data. The Laplace transform has been prevalent in contemporary applications of DP for numerical data.

Definition 1.3.7. *[102] Given a target function f and a parameter $\varepsilon \geq 0$, the random algorithm $A_f(D) = f(D) + x$, where x is a random variable sampled from the Laplace distribution with parameters (μ, Δ_f), is referred to as the Laplace random transformation mechanism. This mechanism satisfies ε-differential privacy (ε-DP).*

Here, Δ_f represents the global sensitivity of the function f, defined as:

$$\Delta_f = \sup |f(D) - f(D')|,$$

where the supremum is taken over all datasets D and D' that differ by at most one element.

The noise addition technique is simple to implement, offering high computational efficiency and minimal communication overhead. As a result, it has become a popular choice in privacy-preserving machine learning applications, including algorithms like Decision Trees [106], Support Vector Machines, and Logistic Regression [107].

The core idea behind differentially private deep learning lies in training a model using raw data while introducing noise-based perturbations at various stages of computation (e.g., gradients during optimization). In deep learning, noise can be added at five critical points: the input, loss function, gradient, training parameters, and output label [108]. When noise is added to data before it is collected, this method is referred to as Local Differential Privacy (LDP). LDP is widely used for aggregating statistical data [109], with notable implementations such as Google's RAPPOR, introduced by Erlingsson et al. [50], to gather user privacy statistics through web browsers.

DP is favored for its efficiency and simplicity compared to other privacy-preserving techniques [110,111]. However, a significant limitation of DP-based models is the inherent trade-off between privacy and accuracy. Achieving strong privacy requires smaller values of the parameter ε which reduces model accuracy. Conversely, increasing ε enhances accuracy but compromises privacy protection.

1.3.3.2. Multiplicative Perturbation

Random projection is a widely used technique in the realm of multiplicative perturbation [112–115]. Certain random projection methods, as explored in [112], have been shown to preserve the dimensionality of data effectively. However, these methods are not without their vulnerabilities; as highlighted in [116], they are susceptible to approximate reconstruction attacks. To enhance privacy, various approaches have been proposed in the literature, focusing on reducing data size while maintaining privacy [113–115]. For instance, the study in [115] implements a standardized projection matrix R applied universally to all participants in the dataset.

Despite its simplicity, this approach raises significant privacy concerns, particularly if participants collaborate to reverse-engineer sensitive data. To mitigate this risk, studies like [113, 114] have proposed using distinct private projection matrices for each participant. While this enhances privacy, it comes at the cost of preserving the Euclidean distances in the perturbed data, leading to a noticeable decline in the performance of distance-based classifiers.

To address this limitation, the methodology in [114] employs regression techniques to reconstruct pairwise distances between original data vectors. This reconstruction leverages publicly shared data samples and their projections to infer distances indirectly. However, a potential privacy risk arises, as the coordinator can exploit the public data and its projections to recover individual random projection matrices. In contrast, the approach in [113] uses deep neural networks (DNNs) to identify complex patterns within the projected data from multiple participants, offering an alternative to regression for estimating distances while enhancing the robustness of the system.

1.3.3.3. Generative Obfuscation

Similar to additive and multiplicative perturbation methods, generative models can produce obfuscated data to mask raw sensitive information. However, the concept of generative obfuscation introduces greater complexity. Huang et al. [117] proposed an innovative approach known as the Generative Adversarial Privacy algorithm. This technique employs a privatizer module in conjunction with an adversary network, forming a framework based on minimax game theory.

In this setup, the privatizer aims to transform the original dataset X in such a way that the adversary network is unable to accurately classify a specific sensitive attribute Y, thereby preserving privacy. The training process for both the privatizer and the adversary is conducted iteratively, with each component refining its strategy to either enhance obfuscation or improve classification accuracy. This dynamic interaction captures the essence of adversarial training and highlights the balance between privacy preservation and adversarial robustness.

1.3.3.4. Data Synthesis

Data synthesis methods leverage generative models to replicate the underlying distribution of sensitive datasets, enabling the creation of synthetic data samples that closely mirror the original data. This approach ensures that the generated data maintains general patterns while protecting information unique to individuals.

An example of this is the application of differentially private k-means clustering, as discussed in [118]. In this method, generative models are trained specifically on data from their respective clusters using differentially private gradient descent. This ensures both privacy preservation and the effective learning of cluster-specific characteristics while adhering to differential privacy guarantees [118].

1.4. An Overview of Research in Privacy-Preserving Deep Learning

The privacy-preserving primitives discussed have shown great utility in traditional machine learning models, including association rule mining, decision trees, k-means clustering, and logistic regression [119]. However, applying these techniques to deep learning models in practical scenarios introduces significant challenges. The inherent complexity of deep learning, with its sophisticated nonlinear functions, makes adapting traditional methods both cumbersome and intricate. Additionally, unique components of deep learning architectures, such as convolutional layers in CNNs and regularization strategies like normalization and dropout, require specialized adaptations when implementing privacy-preserving techniques.

This thesis introduces a classification framework for privacy-preserving deep learning (PPDL) approaches, based on the nature of data exchange during the process. One category focuses on transforming and distributing input data to maintain confidentiality while ensuring that the data can still be used effectively for training and inference. Another approach involves sharing the predictions generated by independently trained models. These predictions are then aggregated to reach a consensus for final outputs, often employing ensemble learning techniques to construct a new model. A third method revolves around distributed learning, where participants collaboratively build a global model by sharing model parameters or gradients derived from their private datasets.

To safeguard the shared data, researchers commonly rely on cryptographic methods, data obfuscation techniques, secure enclave-based strategies, or a combination of these approaches. This framework encapsulates the current methodologies for implementing privacy-preserving mechanisms in deep learning, as visualized in Figure 1.10.

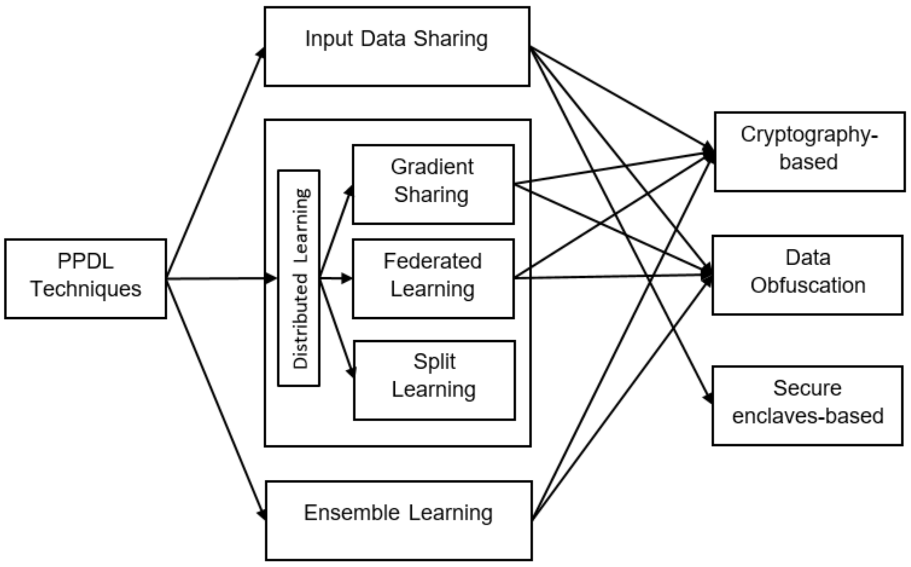


Figure 1.10: A Review of Privacy-Preserving Deep Learning Methodologies

1.4.1. Input Sharing Approach

In this setup, a service provider's server is responsible for both training and deploying the model. User input data is preprocessed using various security protocols, such as SMC and HE, to modify, encrypt, or perturb the data before transmitting it to the server. All training and prediction operations are then performed centrally on the server.

The centralized learning model offers several practical advantages, including:

- This approach significantly reduces the computational rounds on clients, as it eliminates the need for data contributors to actively participate in the training process. Instead, their role is limited to preprocessing and sharing transformed data with the entity responsible for training, simplifying the process and removing the need for any additional steps.

- The approach enables collaboration within a fluctuating network framework, which is especially beneficial in situations where users frequently connect to and disconnect from the network.
- This functionality facilitates executing operations on models with non-identically dispersed data (non-iid).
- This approach enables meaningful collaboration on models even when participants have minimal data, which would otherwise be insufficient for independent model creation. A notable example is fully distributed data models, where each participant holds only a single data point [120].
- The shared data serves as a foundation for building models that can address a wide variety of purposes and challenges. For instance, a centralized training server can utilize the same dataset to develop a credit scoring system, evaluate insurance needs, detect financial fraud, and perform other related tasks, offering versatile applications to all users.

Thanks to its inherent advantages, this model has gained substantial attention in empirical research. In this framework, data transmission is initiated by users, while the server performs all computational tasks. The interaction between participants and the server revolves around sharing user input data, with a strong emphasis on safeguarding confidentiality through input transformation strategies.

Data collected from participants is first transformed and then transmitted to the server for computation and training purposes. This transformation process ensures that sensitive information remains secure before being shared [121, 122]. Commonly used preprocessing techniques include HE algorithms [123, 124], SMC protocols [125], secret sharing schemes [51], and noise addition methods [126].

The Data Sharing Approach, a method extensively utilized in research on PPDM [127] and PPML [128], allows for its application during both the training and prediction stages. Despite its advantages, this method faces significant challenges, particularly concerning its efficiency and the precision of outcomes.

One major issue is the high communication and computational overhead involved in distributing large datasets among participants for training, rendering this approach impractical for deep learning privacy measures [129]. Additionally, methods that transform data often involve a compromise, introducing or permitting noise that negatively affects the accuracy of training and prediction [126, 130]. In cryptographic approaches, SMC necessitate conversion to large numerical values, necessitating adjustments for deep neural network operations to accommodate these large integers. For instance, approximations of nonlinear functions in neural networks have been proposed using polynomial functions to adapt to these requirements [131].

Further adaptations include Chabanne et al.'s substitution of the ReLU activation function with a quadratic function and opting for sum-pooling layers over max-pooling to suit cryptographic constraints [132]. Similarly, nonlinear function replacements with Chebyshev polynomial approximations have been explored to maintain model functionality under these constraints [133]. However, these modifications compromise model accuracy and adaptability. Cryptographic security measures demand operations on large integers, inflating data sizes and computational demands, and diminishing model performance [134]. The necessity for secure key exchange or distribution among participants also adds to the computational and communication burdens.

In contrast, non-cryptographic models employing data perturbation or artificial data generation offer lower computational and communication costs and greater versatility across different network architectures and models [135]. However, the trade-off between security and accuracy limits their effectiveness, as incorporating noise to protect data privacy invariably impacts model accuracy. While minimizing noise can preserve accuracy, it also leaves the model vulnerable to attacks aimed at inferring or reversing the data.

1.4.2. Output Sharing Approach

The Private Aggregation of Teacher Ensembles (PATE) method offers a privacy-centric strategy for deep learning by focusing on the use of aggregated insights rather than direct data sharing [136]. In this model, individual participants, referred to as ”teachers,” train their models independently on their datasets. Instead of sharing these datasets, they transmit prediction results to a central ”student” or aggregation server. This server then employs ensemble learning to derive general predictions from these results and applies additional training techniques to refine models further. A key benefit of the PATE approach is its flexibility; it doesn’t necessitate uniform model structures or identical hyperparameters among participants, and it safeguards model confidentiality to protect intellectual property by preventing disclosure of network structures or hyperparameters.

However, PATE can significantly reduce accuracy, requiring a balance between privacy and performance, even for simple tasks like MNIST. Scalable PATE [137] and other enhancements have attempted to address these limitations by improving confidence in the teachers’ consensus and integrating novel methods such as weighted ERM [138] and individualized privacy [139] adjustments to preserve data utility while maintaining privacy.

To overcome the challenges of high-dimensional data, PATE-GAN [140] and G-PATE [141] have been developed to train generative models under the PATE framework, enhancing data utility for machine learning applications without compromising privacy. Moreover, adaptations like PATE-AAE [142] for voice classification and SeqPATE [143] for text generation show the versatility of PATE in handling diverse data types and complex model outputs.

Despite significant advancements, data obfuscation techniques often come with trade-offs, such as a reduction in model accuracy, and they remain susceptible to threats like model inversion attacks, which pose persistent privacy concerns. While the integration of cryptographic methods such as homomorphic encryption (HE) and secure multi-party computation (SMC) has enhanced security, these approaches introduce substantial computational overhead. This underscores the ongoing need for more efficient and scalable privacy-preserving solutions in practical applications.

Integrating cryptography into frameworks like PATE has notably strengthened their security measures. Techniques such as HE and SMC enable the aggregation of teacher models' outputs without exposing sensitive information, effectively mitigating the risk of information leakage. These cryptographic safeguards ensure that even if aggregated data is compromised, attackers cannot reconstruct the private training data.

For instance, the SPEED framework applies noise to the teachers' outputs and employs homomorphic encryption to ensure the aggregator cannot infer sensitive information while still allowing for necessary computations [144]. Similarly, the CaPC learning platform combines HE, SMC, and private aggregation methods to facilitate secure and confidential collaborative learning across multiple parties. This approach preserves privacy without compromising the flexibility and utility of the machine learning models involved [145].

These cryptographic techniques, however, introduce new challenges. The computational cost of processing encrypted data can be substantial, particularly for complex machine learning models and large datasets. This computational burden limits the practical deployment of such privacy-preserving methods, especially in scenarios where processing power is constrained.

Moreover, while these methods provide robust security against external and non-colluding adversaries, they are less effective if insiders or collaborators decide to breach privacy protocols. For instance, in systems where the aggregator and student can collaborate, they may potentially exploit the system to access sensitive teacher data, posing a significant insider threat.

Overall, while cryptography offers a powerful tool for enhancing privacy in machine learning, it must be balanced with considerations of computational efficiency and potential insider risks to be viable in real-world applications.

1.4.3. Model Sharing Approach

The third method, known as model sharing or distributed learning, has proven to be highly effective and is widely used in various applications, including Google Keyboard [146]. In this approach, participants independently train models on their own local data and then share the resulting intermediate parameters or gradients with a central aggregation server or directly among each other.

Model sharing addresses several key challenges found in earlier methods. First and foremost, it minimizes the need for large-scale data transfers, as only the model parameters or gradients—much smaller than the raw data—are communicated. This significantly reduces both communication costs and bandwidth demands. Furthermore, because the raw data never leaves the local environment, the privacy of sensitive information is maintained, as no direct exposure of private data occurs. This approach includes techniques such as split learning [147] and federated learning (FL) [148], which are designed to safeguard data privacy while enabling collaborative model training.

1.4.3.1. Split learning

In split learning [147], as outlined, participants train portions of a model up to a certain layer before forwarding the weights to a central server, which completes the training. This process, which alternates between client and server based on back-propagation, limits the direct exposure of sensitive data.

Transfer learning has been adapted into a hybrid approach, allowing clients to pre-train certain layers of a model before the cloud server finalizes the training [149, 150]. This method splits the model into segments that separately handle sensitive data on the client side and more generic training on the server side, thus enhancing data privacy.

The efficacy of these approaches hinges on the complexity of inverting the transformed data back to its original form, especially through layers of non-linear activation functions, making direct data reconstruction challenging. This model decomposition method requires less bandwidth than other distributed learning strategies and mitigates the risk of indirect data leakage, despite potential accuracy reductions due to limited shared information.

1.4.3.2. Federated learning

The federated learning protocols operate within a network of nodes, denoted as $\mathscr{U} = U_1, U_2, \ldots, U_n$, where each node, U_i, maintains a unique private dataset, D_i. The goal of this collaborative framework is to develop a shared global model while safeguarding the privacy of each node's data. The process begins with a central server initializing a global model, W^0, and distributing key hyperparameters, such as the number of local epochs E and the batch size $|B|$, to all participating nodes.

At each communication round, indexed by t, the nodes update the current model, W^t, using their local datasets, following the specified parameters E and $|B|$. After training, each node sends its updated model, W_i^t, to the server, where the models are aggregated to form a new global model, W^{t+1}. This updated model is then redistributed to all nodes, continuing the iterative training process. The federated learning framework operates in a cyclical manner, ensuring constant refinement of the global model while preserving the privacy of the raw data at each node.

- **Selection and Deployment of the Global Model:** The process begins with the selection of a robust pre-trained machine learning (ML) model, referred to as the "**global model**," which is equipped with initial parameters. This model serves as the foundation that is distributed across the FL client network, initiating the collaborative learning process.
- **Local Training:** Each client, armed with the global model, conducts local training using their own private dataset. This localized training leads to the development of individual model versions, enriching the overall learning process through diverse data sources and personalized updates.
- **Aggregation and Refinement of Knowledge:** The cycle culminates when each client sends their locally trained model updates back to the central server. The server then aggregates these updates, refining and enhancing the global model by combining the knowledge from all clients. Once integrated, the updated global model is redistributed to all participants, restarting the cycle with improved intelligence.

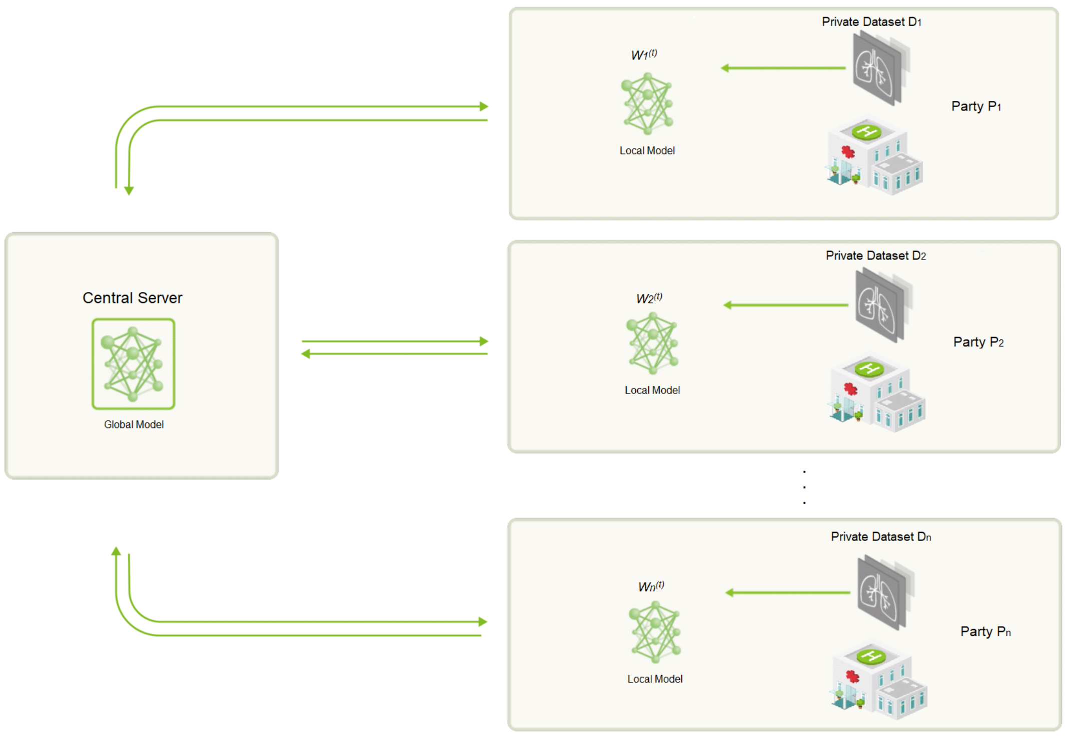


Figure 1.11: Federated Learning Framework

This process forms an ongoing cycle of continuous refinement, where the global model remains constantly updated and synchronized with the evolving data from each client. A visualization of the federated learning architecture is shown in Figure 1.11.

Expanding on this idea, McMahan et al. [3, 4] introduced the Federated Averaging (FedAvg) algorithm, which refines the federated learning (FL) process. In this framework, training data is kept on local devices, and only the model parameters are shared with the central server. The server then aggregates these parameters to update the global model, which is periodically redistributed to participants for further training. By reducing the frequency of communication between the clients and the server, FedAvg helps mitigate the risk of data leakage and alleviates the strain on network resources.

However, the averaging process, while efficient, can compromise model performance, particularly when data is unevenly distributed across clients or in the presence of network instability. The authors also point out that this approach places a significant computational burden on participants, as it requires complete local processing of the data. To address the substantial bandwidth demands, data compression techniques have been proposed. Despite its advantages, the model's lack of specific security measures when sharing parameters raises concerns about potential vulnerabilities, particularly from inversion or inference attacks [8].

Rather than directly sharing local model parameters, some training approaches, such as Stochastic Gradient Descent (SGD) with large aggregate batch sizes, focus on sharing gradients calculated from one or more batches of participants. These gradients are aggregated to form a global gradient, which is then used to update the global model's parameters. This approach ensures high accuracy, as the parameter updating process closely mirrors the standard deep learning training method [151, 152]. However, the process requires substantial data exchange between clients and the server, leading to significant network resource consumption, which can be even higher than in other sharing techniques. Additionally, since gradients are shared in plaintext, there is a risk of data leakage.

To address this privacy concern, Reza Shokri et al. [153] proposed the concept of *selective learning*. In this model, participants train their local models based on their respective datasets and then share selected portions of their gradients with the aggregation server. Rather than transmitting the entire gradient vector, participants only share a fraction of it, which helps protect privacy by making it more difficult for an attacker to extract sensitive information. The server then updates the global model's parameters using the SGD algorithm, and the updated model is sent back to all participants. This selective sharing significantly improves privacy protection, but it does come with trade-offs. The reduced gradient information can negatively impact the accuracy of the model, and the sequential nature of the parameter updates results in high latency. Additionally, the selective learning model does not support concurrent execution among participants, further exacerbating the delay. Despite these measures, some residual information may still be vulnerable to attacks, allowing potential adversaries to infer data from individual clients [23].

To mitigate these risks, researchers have turned to data obfuscation and cryptographic techniques, which protect the parameters and gradients before they are shared with others.

1.4.3.3. Data Obfuscation techniques

Abadi et al. introduced a method for enhancing privacy in parameter sharing by incorporating noise into the gradient vector based on Gaussian or Laplace distributions, a technique further expanded upon in various network architectures to safeguard privacy [154]. Additionally, Truex et al. developed the LDP-Fed federated learning system, which employs local differential privacy (LDP) to provide formal privacy assurances [155]. Given that LDP traditionally focuses on discrete datasets, adapting it to the continuous, high-dimensional data typical in federated learning presented challenges. LDP-Fed addresses this by ensuring differential privacy during the iterative collection of training parameters across extensive neural networks, supplemented by selective parameter update strategies to maintain privacy.

This approach necessitates a balance between model accuracy and privacy protection. To mitigate the impact on accuracy, enhancements like normalization and gradient clipping have been implemented, boosting efficiency with minimal computational and communication demands [154]. Despite these improvements, the introduction of significant noise can adversely affect model precision. While reducing the likelihood of reverse or interference attacks, this method remains susceptible to Generative Adversarial Network (GAN) attacks [23], illustrating the ongoing trade-offs in privacy-preserving techniques in machine learning.

1.4.3.4. Cryptography-based techniques

In federated learning systems, the aggregation algorithm plays a pivotal role by merging the updates from local models, provided by various clients, into a singular, improved global model. Google's federated learning framework employs the Federated Averaging (FedAvg) algorithm [3], which designates the central server as the orchestrator of the training process. This server initially disseminates the global model and its parameters to a select subset of clients, or a "mini-batch". These clients proceed to train the global model locally with their data, after which they return their model's weights to the server. The central server then integrates these local updates into the global model by performing a weighted averaging of all the local model updates. This process effectively combines the learning from across the network to refine and update the global model.

$$W^{t+1} \leftarrow \sum_{i=1}^{n} \frac{m_i}{M} W_i^t \tag{1.4.6}$$

In this context, m_i represents the number of data records held by party U_i, M denotes the total number of records across the entire system, and $W_i^t \in \mathbb{R}^{model\_size}$ refers to the model parameter vectors of party U_i at the global epoch t.

The configurable criterion, specifically the number of training rounds, serves as the stopping condition for the coordinator to conclude the training rounds and proceed with averaging the local model updates.

Various adaptations of FedAvg exist, such as FedProx [156], FedMA [157], Scaffold [158], and FedBCD [159], each tailored to optimize specific parameters within the implementation of Federated Learning. However, in practical applications, FedAvg consistently demonstrates notably effective performance, often surpassing or at least matching the performance of other variants across a range of Federated Learning scenarios [160].

The primary objective of this thesis is to compute the weighted average of parameter vectors in a shared deep learning model among participating parties denoted as U_i within the FedAvg framework. This computation is based on Formula 1.4.6 above. The number of samples each participating party holds is assumed to be predetermined, and the total number of data records remains constant. In essence, the central problem addressed here is to aggregate the parameter vectors securely:

$$W^{t+1} \leftarrow \sum_{i=1}^{n} W_i^t \tag{1.4.7}$$

The current challenge revolves around resolving the secure vector sum problem:

$$V = \sum_{i=1}^{n} W_i^t \tag{1.4.8}$$

The thesis aims to calculate the total vector V without disclosing the individual vectors W_i^t of the participating parties at each global epoch t.

Further research in PPDL is dedicated to addressing this multi-party secure computation challenge. The SMC protocols designed to safeguard the parameter aggregation process in Federated Learning are primarily categorized into two distinct techniques: secret sharing and homomorphic encryption.

a. Secure Multi-party Sum Protocols based on secret sharing techniques

The secret-sharing technique is the simplest among various secure multiparty computation techniques. It enables parties to share their secret values into several parts for computations and then aggregate them to obtain the final calculation result. In the context of the federated learning model, where secret values are the parameters of the local training model at each step of the participants, the application of secret-sharing techniques is often calibrated and optimized to ensure the efficiency of the computation process.

Hosseini et al. [161] provide a technique for randomly dividing the model's parameters and distributing the resulting weights multiplied by the model's parameters among other participants. This strategy is the most straightforward technique for utilizing the secret sharing mechanism in PPDL. In the context of a multiparty system, the protocol incurs significant expenses in terms of transmission and processing. This procedure is appropriate for systems with a limited number of players.

The notion of secure aggregation in the context of FL was first presented by Bonawitz et al. in their influential publication [162]. The protocol is designed to endure client disconnections and utilizes obfuscating, unpredictable values, Shamir's Secret Sharing (SSS) [163], and symmetric encryption to prevent unwanted access to local models. Nevertheless, the process of aggregating necessitates a minimum of four instances of interaction between each client and the aggregator during each cycle, which can be burdensome for clients, particularly those connected via a wide area network (WAN) with restricted resources. Moreover, the protocol's significant communication overhead and complexity, combined with its limitations around handling real-number data, complicate its practical application in real-world parameter sharing scenarios, underscoring the ongoing challenges in achieving both efficient and secure privacy-preserving machine learning methods.

The protocols utilized by VerifyNet [164] and VeriFL [165] are derived from the protocol introduced by Bonawitz et al. [162]. Both protocols augment the verifiability feature of the current protocol [162] to guarantee the aggregation process's precision. Nevertheless, they rely on a trustworthy entity to generate public/private key pairs for all participating clients.

In recent studies, researchers in [166] and [167] have proposed secure aggregation protocols that exhibit polylogarithmic communication and computing complexity. These protocols effectively minimize the overhead compared to the approach presented in [162]. The primary concept proposed by the authors is to substitute the star topology of the communication network as described in reference [162] with random subgroups of clients. Additionally, they suggest employing secret sharing exclusively for a subset of clients rather than for all pairs of clients. Both methodologies necessitate three iterations of communication between the server and clients.

FastSecAgg [168] offers a safe aggregation mechanism that relies on the Fast Fourier Transform multi-secret sharing technique. The protocol exhibits robustness against adaptive attackers, who can adaptively corrupt clients during the execution of the protocol. FastSecAgg is a three-round interactive protocol designed for private federated learning.

The Turbo-Aggregate method, as described in reference [169], effectively mitigates the burdens of communication and computation associated with secure aggregation compared to the approach presented in reference [162]. This method employs a circular communication topology to achieve its objectives. The primary limitation of Turbo-Aggregated lies in its round complexity, which is $O(\frac{n}{\log n})$, where n is the number of updates or clients involved.

The SAFER framework [170] aims to minimize communication expenses in federated learning (FL) by implementing update compression techniques. Additionally, it incorporates a secure aggregation protocol that relies on arithmetic sharing to ensure data privacy and security. Nevertheless, SAFER exclusively considers training sessions involving fewer than ten customers and excludes any instances of clients discontinuing the program. Furthermore, the SAFER model was exclusively evaluated on datasets that follow the independent and identically distributed (IID) assumption. Consequently, it remains to be seen if SAFER can effectively handle the non-IID data commonly employed in federated learning (FL) scenarios.

Tran et al. [171] split model parameters into various parts and sent each to a group of members. This method can enhance privacy and reduce the communication and computation cost of the model. This protocol ensures relatively high security because any group of members can only receive a part of the shared parameters, so recovering all the local model parameters will require the participating members to all collude. In addition, sending only a part to a few members significantly reduces the communication and computational costs needed compared to sharing the entire model. However, each participant needs to send his or her parameter set to a group of other members before sending it again to the aggregation server, making this protocol's computation and communication costs still very high.

In [172], the authors proposed CE-Fed to secret share the model parameter to reduce the communication cost incurred in traditional MPC-enabled Federated Learning. The CE-Fed, as suggested, designates a limited number of clients to serve as committee members. These committee members utilize the MPC service to consolidate the local models of all FL clients hierarchically. As a result, it circumvents the dissemination of model parameters from individual clients to all other participants on the Federated Learning roster. The suggested CE-Fed is implemented using a two-phase approach. During the initial stage, the FL clients in close proximity are organized into groups. The models of FL clients within a given group are securely aggregated using MPC to create an intra-group model. The aggregation committee is formed by selecting one client from each group based on latency. During the subsequent stage, the committee members collaborate to consolidate the inter-group models utilizing secret sharing techniques.

In [173], Tran et al. proposed an effective FL protocol based on secret sharing. Instead of sharing secret parameters with many parties, they add masking noises and send them twice over 2 phases through different anonymous channels. These parameters' origin values are then hidden from any other party. Thus, the protocol ensures privacy while reducing the number of communication phases significantly. However, this protocol has the disadvantage of requiring the participants to be highly stable and not allowed to leave the training process at any time. In addition, the latency due to the anonymous channel and the security of the anonymous channel are also some issues to consider.

The authors of [174] and [175] use additional cryptographic algorithms and DP to improve the security of secret parameter sharing. The authors have thoroughly analyzed different attack cases and their effects on the FL model and proposed using cryptographic solutions and DP to solve these problems. However, the proposed protocol is also not resistant to the model in which the parties collude with the aggregator server.

b. Secure Multi-party Sum Protocols based on encryption

To enhance the privacy of local models beyond differential privacy (DP), various cryptographic tools have been suggested. Phong et al. [176] introduced a scheme using Learning With Errors (LWE) homomorphic encryption to secure gradient vectors exchanged between clients and a semi-trusted aggregation server. This server conducts addition calculations solely on the encrypted data, thus avoiding the need for decryption. All computations are performed directly on the encrypted data, ensuring that the values of the gradient vectors remain concealed. Post-computation, the resulting encrypted output is returned to the clients for decryption, enabling subsequent round calculations. This protocol establishes a secure environment devoid of collaboration between any individual participant and the aggregator. This method, however, necessitates that all participants be semi-trusted entities possessing a common decryption key to access the model. The privacy protection breaks down if any participant decides to work with the server, highlighting a potential vulnerability.

However, a notable limitation of this protocol and all SMC protocols employing HE is that any collusion may lead to the aggregation server gaining access to decryption keys. Consequently, the security of the gradient vector values could be compromised. This drawback is significant in this protocol and the broader scope of SMC protocols that rely on HE.

Truex et al. [177] employ a combination of additive HE and DP. However, their approach is not resilient to client dropouts. The utilization of HE incurs a notable increase in runtime execution time, and their system necessitates three successive rounds of communication. These factors render the application of FL in real-world scenarios unfeasible.

The EaSTFfly framework [178] uses either additive homomorphic encryption (HE) with packing or Shamir's secret sharing (SSS) [163] in conjunction with quantization. Instead of utilizing FL's FedAvg mechanism [162], the clients share their gradients following each training iteration, resulting in a notable increase in the total number of training iterations. The application of FedAvg is hindered in scenarios where additive HE or SSS prevents the division operation from being performed. Moreover, EaSTFfly's HE protocol requires all clients to possess the same secret key. All updates can be decrypted if there is a collision between a client and the aggregator.

The technique, BatchCrypt [179] effectively decreases the encryption and communication overhead associated with aggregation based on Homomorphic Encryption (HE). This is achieved by utilizing a batch encryption method, which requires only one round of communication. Once again, the utilization of costly hardware accelerators (e.g., [177], [178]) renders it impractical to employ for real-world training in the context of federated learning.

The HybridAlpha framework, described in reference [180], incorporates functional encryption and differential privacy techniques. Functional encryption involves deriving public keys for all clients from a private/public master key pair. The proposed method enhances the runtime of [177] by a factor of 2× and exhibits tolerance toward dropouts. Nonetheless, HybridAlpha depends on a reliable entity that possesses the primary keys and routines control over the manipulation of aggregation weights by the aggregator.

The encryption method employed by POSEIDON [181] encompasses the entire FL process, including the local training conducted by the clients. Consequently, this introduces a substantial computing burden on the devices of each client. The authors propose modifying the existing communication structure for clients, advocating for adopting a tree-like network configuration instead of the conventional star topology. In the proposed network, clients are interconnected hierarchically, with communication occurring through intermediate nodes rather than direct communication with a central aggregator. Furthermore, utilizing a distributed bootstrapping technique effectively updates ciphertexts while implementing an alternating packing strategy improves the efficiency of training neural networks while preserving encryption. However, the support for clients' dropouts in POSEIDON is limited to cases when decentralized bootstrapping is not utilized.

The SAFELearn framework [182] is implemented by the integration of various protocols in Multi-Party Computation (MPC) and Secure Two-Party Computation (STPC). Specifically, Boolean sharing is employed to evaluate the Argmin operation securely, while arithmetic sharing is utilized to evaluate multiplication and addition operations securely.

As referenced in [44]–[47], various studies incorporate secure computation methodologies and differential privacy into machine learning approaches. Notably, these designs are often customized for specific machine learning (ML) algorithms without a specific emphasis on federated learning (FL). Consequently, they may not effectively address scenarios involving client dropouts, may not scale efficiently when dealing with a large number of clients, might employ costly cryptographic techniques, and may not be suitable for dispersed training scenarios.

1.5. Comparison of the PPDL Approach and Existing Limitations

Drawing from the analysis in the preceding sections, Table 1.1 outlines the distinct features, advantages, and limitations of various methodological approaches. The approach of input sharing, designed to bolster security, typically necessitates the incorporation of techniques such as noise addition or cryptography. However, this approach's reliance on noise addition tends to compromise its security, rendering it susceptible to data inference attacks, and significantly diminishes the model's accuracy due to alterations in the statistical properties of the data. On the other hand, integrating SMC enhances security but at the cost of accuracy, as adapting deep learning models to cryptographic primitives presents substantial challenges. This integration also imposes considerable computational and communication demands owing to the complexity of both the data and the models. Additionally, the necessity for key sharing among participants means that the model's security is guaranteed only in scenarios involving two-party computation (2PC); in more collaborative models with numerous participants, security assurances wane, rendering it predominantly suitable for the prediction phase rather than the training phase.

Methods	Privacy	**Utility Reduction**	**Cost**	**Requirements**
Input sharing	-	No	Low	-
Input sharing + SMC	High	High	High	- Share the same keys - Convert model and params
Input sharing + DP	Medium	High	Low	NO
Output sharing	Low	High	Low	- Local models have good performance - Public dataset
Output sharing + DP	Medium	High	Low	- Local models have good performance - Public dataset
Output sharing + SMC	High	High	High	- Local models have good performance - Public dataset
Split Learning	Medium	High	High	- 2 parties
FL	Low	Low	Low	NO
FL + SMC	High	Low	Medium	NO
FL + DP	Medium	High	Low	NO

Table 1.1: Comparing PPDL approaches

The method of output sharing has a notable effect on model performance, primarily due to prediction errors that arise when passing data through teacher models. While this approach offers great flexibility, it relies heavily on access to public data and high-quality local models, which can be challenging to obtain, especially in scenarios where data is distributed and participants have limited data availability.

The focus then shifts to the ”model sharing” strategy, which can be further divided into split learning and federated learning. Split learning, which involves sharing parameters only across specific layers of the network, is constrained by a limited number of participants and often suffers from accuracy degradation due to potential information leakage. In contrast, federated learning proves to be the most effective and practical solution for training distributed deep learning models across multiple parties. It successfully mitigates direct data leakage, strikes a balance between model accuracy and computational efficiency, and allows for parallel processing. However, it is still susceptible to indirect data leakage through the exposure of model parameters. To address these risks, techniques like DP and SMC are proposed. While DP offers enhanced privacy at the cost of some accuracy—potentially leaving the system vulnerable to attacks such as inversion through methods like GANs—the combined use of Federated Learning and SMC presents a promising research avenue. This combination can help preserve both the integrity and accuracy of the model, and the thesis aims to further investigate this approach.

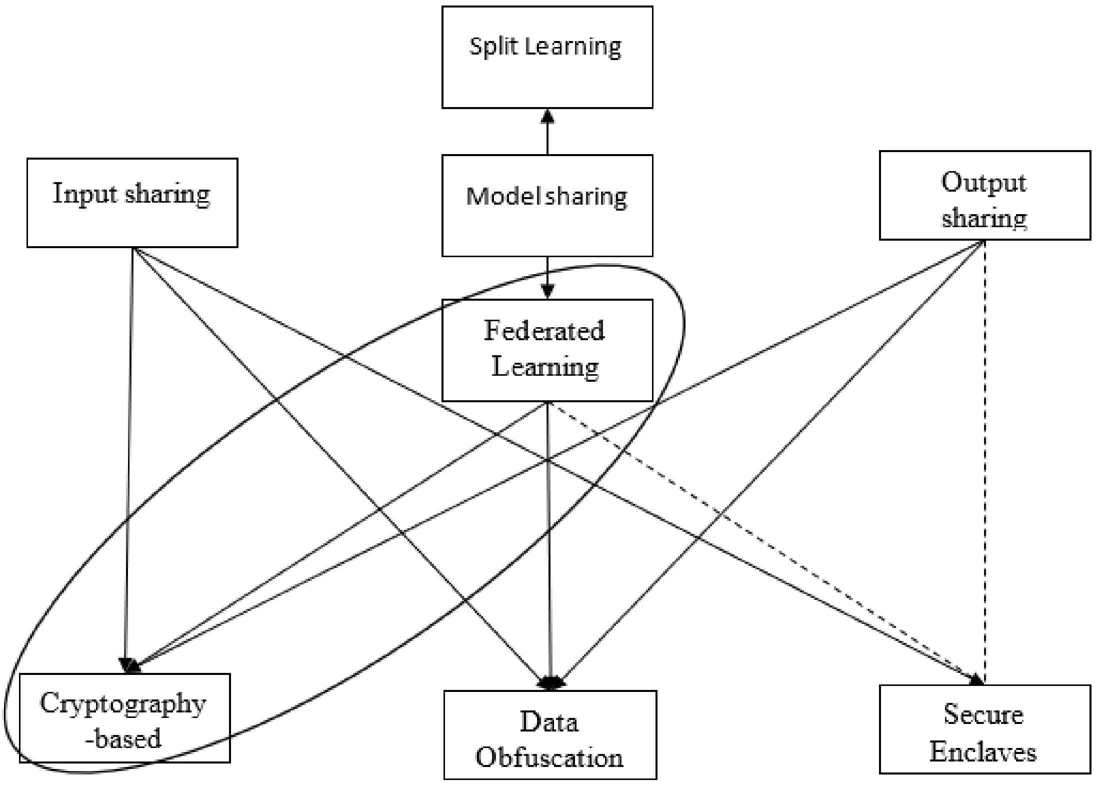


Figure 1.12: Thesis’s Objective

Nevertheless, the integration of FL and SMC presents several significant challenges:

- Existing models under development require participants to either directly exchange cryptographic keys or rely on a trusted intermediary for key distribution. This security model, however, is susceptible to compromise if participants collude, either among themselves or with the central aggregation server.
- Additionally, these models often involve converting real numbers into large integers, a process that notably increases both computational overhead and the time required for calculations and data transmission.

Therefore, this thesis proposes the development of secure aggregation protocols for real-valued vectors. The goal is to ensure the safety and efficiency of the parameter aggregation process in Federated Learning while addressing the aforementioned challenges, striking a balance between security, performance, and practicality.

1.6. Chapter Summary

This chapter has provided an overview of the privacy challenges in deep learning and current solutions. It emphasizes Federated Learning as a promising approach for protecting privacy during deep neural network training, while cryptographic techniques show potential for secure parameter sharing. However, these approaches face two key issues: key sharing, which risks collusion, and the handling of real numbers, which requires encoding and may reduce model accuracy.

To address these challenges, this dissertation proposes new training protocols for distributed deep learning networks, incorporating SMC for enhanced security and efficiency, even in the presence of collusion. The innovative methods developed will be discussed in Chapters 2 and 3. The work in this chapter is published in **Publication 1**.

CHAPTER 2. PROPOSING SOME FLOATING POINT REAL NUMBER SECURE MULTI-PARTY VECTOR SUM PROTOCOLS

This chapter starts with some basic concepts of Cryptography used to construct the proposed protocol, specifically Elgamal and Elliptic Curve Cryptography (ECC). Following that, the subsequent sections sequentially introduce and detail three novel SMC protocols developed for privacy-preserving federated learning. These protocols include: secure multi-party vector summation with floating point real numbers using integer quantization, secure multi-party summation of floating point real number vectors with a masking matrix and Elliptic Curve Cryptography (ECC), and secure multi-party summation of floating point real numbers without the need for pre-established secure/authenticated channels. These proposed protocols are documented in **Publications 3, 5, 6, and 7**.

2.1. Cryptography preliminaries

2.1.1. Discrete logarithm problems

In cryptographic protocols, the discrete logarithm problem plays a crucial role. This section explores the key concepts underlying this problem, as discussed in [183]. Let $\mathbb{G}$ represent a cyclic group of order q. This means that for any element $h \in \mathbb{G}$, there exists a unique value $x \in \mathbb{Z}_q$ such that $g^x = h$. In this setting, the expression $x = \log_g h$ denotes that x is the discrete logarithm of h with base g. The computationally difficult discrete logarithm problem is formally defined as follows:

Definition 2.1.1. *Let $\mathbb{G}$ be a cyclic group of order q, generated by an element g, and let $h \in \mathbb{G}$ be a chosen element. The discrete logarithm problem in $\mathbb{G}$ involves determining the value of $\log_g h$, which is the exponent x such that $g^x = h$.*

The discrete logarithm problem is considered hard in $\mathbb{G}$ if, for any probabilistic polynomial-time algorithm A, the probability of A successfully solving the problem is negligible. In other words, even with access to $(\mathbb{G}, q, g, h)$, the likelihood of A finding an $x \in \mathbb{Z}_q$ such that $g^x = h$ is extremely low.

This difficulty also underpins other problems such as the Computational Diffie-Hellman (CDH) and Decisional Diffie-Hellman (DDH) problems.

- **Computational Diffie-Hellman (CDH) problem**

Given the parameters $(\mathbb{G}, q, g)$ and two elements $h_1 = g^{x_1}$ and $h_2 = g^{x_2}$ in $\mathbb{G}$, the operation $DH_g(h_1, h_2)$ is defined as $g^{x_1 x_2}$. The Computational Diffie-Hellman (CDH) problem involves computing $DH_g(h_1, h_2)$ from h_1 and h_2.

It is clear that if the discrete logarithm problem in $\mathbb{G}$ can be easily solved, then the CDH problem can also be solved. However, the reverse is not necessarily true; the difficulty of the CDH problem does not imply that the discrete logarithm problem is also hard. As a result, the CDH assumption is rarely used in cryptographic contexts.

- **Decisional Diffie-Hellman (DDH) problem**

Given the parameters $(\mathbb{G}, q, g)$ and three elements $X = g^x$, $Y = g^y$, and $Z = g^z$, where x, y, and z are randomly chosen from $\mathbb{Z}_q$, the hard decisional Diffie-Hellman (DDH) problem is formally defined as follows:

Definition 2.1.2. *The decisional Diffie-Hellman (DDH) problem is considered hard with respect to $\mathbb{G}$ if, for all probabilistic polynomial-time algorithms A, there exists a negligible function $\mu(n)$ such that*

$$\left|\Pr\left[A\left(\mathbb{G}, q, g, g^x, g^y, g^z\right) = 1\right] - \Pr\left[A\left(\mathbb{G}, q, g, g^x, g^y, g^{xy}\right) = 1\right]\right| < \mu(n)$$

This definition asserts that the tuples (g^x, g^y, g^z) and (g^x, g^y, g^{xy}) are computationally indistinguishable, where x, y, and z are randomly selected from $\mathbb{Z}_q$. In essence, the difficulty of solving the DDH problem implies that no polynomial-time algorithm can reliably distinguish between these two tuples with a probability significantly greater than a negligible function $\mu(n)$.

Thus, the hard decisional Diffie-Hellman (DDH) assumption remains a foundational and widely used assumption in modern cryptography.

2.1.2. ElGamal public-key cryptosystem

This section presents a well-known variant of the ElGamal encryption scheme [184], which is based on the hardness of discrete logarithm problems. Due to its cryptographic strengths, ElGamal encryption is an essential building block for the proposals in this thesis and the broader field of SMC.

Consider a cyclic group $\mathbb{G}$ of prime order q, where computing discrete logarithms is computationally difficult. Let g be a generator of $\mathbb{G}$, x be a private key uniformly chosen from $\{1, 2, \ldots, q-1\}$, and the corresponding public key be $h = g^x$.

During the encryption phase, the sender uses the public key h to generate the ciphertext C from the plaintext message m. The process involves selecting a random k from $\{1, 2, \ldots, q-1\}$, and computing the ciphertext $C = (C_1 = mh^k, C_2 = g^k)$. To decrypt the ciphertext C, the receiver uses their private key x to compute the plaintext message as $m = C_1(C_2^x)^{-1}$.

Under the DDH assumption, ElGamal encryption ensures semantic security, making it resistant to chosen-plaintext attacks. This cryptosystem has been widely used to build secure cryptographic protocols, including the ElGamal digital signature [184] and the Schnorr signature scheme [185].

Moreover, ElGamal encryption possesses homomorphic properties, which are critical in the design of SMC protocols.

- *Multiplicative homomorphic property:* ElGamal encryption supports multiplicative homomorphism. Specifically, if $C(m_1) = (m_1 h^{k_1}, g^{k_1})$ and $C(m_2) = (m_2 h^{k_2}, g^{k_2})$ are the ciphertexts of messages m_1 and m_2, then their product $C(m_1)C(m_2) = (m_1 m_2 h^{k_1+k_2}, g^{k_1+k_2})$ corresponds to the ciphertext of the product $m_1 m_2$.
- *Additive homomorphic property:* When the plaintexts are not excessively large (as in typical SMC applications), ElGamal encryption also exhibits additive homomorphism. If the ciphertexts of m_1 and m_2 are given by $C(m_1) = (g^{m_1} h^{k_1}, g^{k_1})$ and $C(m_2) = (g^{m_2} h^{k_2}, g^{k_2})$, then the product $C(m_1)C(m_2) = (g^{m_1+m_2} h^{k_1+k_2}, g^{k_1+k_2})$ represents the ciphertext of $m_1 + m_2$.

These homomorphic properties make ElGamal encryption an invaluable tool for secure computations in the SMC context, where operations on encrypted data are essential for maintaining privacy and security.

2.1.3. Elliptic Curve Cryptography

Additionally, an elliptic curve variant of the ElGamal cryptosystem is proposed in [186], which can be described as follows:

Let $E(\mathbb{F}_q)$ denote an elliptic curve defined over a finite field $\mathbb{F}_q$, with a point O representing infinity. Assume that q is a large prime, ensuring the intractability of the elliptic curve discrete logarithm problem. Let G be a base point on the curve E with order q, meaning that $qG = O$.

The private key d is a randomly chosen integer from $[1, q-1]$, and the corresponding public key Q is computed as $Q = dG$, where d is the private key.

In the encryption process, the sender uses the public key Q to generate the ciphertext C corresponding to the plaintext m. This involves selecting a random integer k from $[1, q-1]$, and computing the ciphertext components as $C = (C_1 = P_m + kQ, C_2 = kG)$, where P_m is a point on the elliptic curve such that the x-coordinate of P_m encodes the plaintext message m.

To decrypt the ciphertext C using the private key d, the receiver computes $m = x_M$, where $M = C_1 + (-dC_2)$, and extracts the plaintext from the x-coordinate of the resulting point M.

As with the classical ElGamal encryption, this elliptic curve variant also guarantees semantic security, provided the decisional Diffie-Hellman (DDH) assumption holds for the elliptic curve E. This ensures that the cryptosystem is resistant to chosen-plaintext attacks, maintaining a high level of security for elliptic curve-based cryptographic protocols.

2.1.3.1. Solving discrete logarithm problems with small space of solutions

The ElGamal and ECC are widely regarded as secure due to the inherent difficulty of solving discrete logarithm problems. However, when the ciphertexts are confined to a small range, exhaustive search techniques can be used to recover the plaintext. This characteristic is particularly relevant in the context of SMC protocols.

For smaller plaintexts, such as m, deriving values like g^m or mG is computationally feasible without requiring significant resources. Algorithms like Shanks' baby-step giant-step method [187] are effective in solving these computations in an efficient manner. However, as the size of the plaintext m increases, the difficulty of solving such problems grows substantially, making the process far more challenging.

- **Notation**

In the subsequent sections, this dissertation will adhere to the notations outlined in Table 2.1, which serve as the foundational symbols and terminologies throughout the analysis. These notations provide a consistent framework for discussing key concepts and computational models, ensuring clarity and precision in the presentation of the research findings.

2.2. Secure Multi-Party Vector Sum Protocol with Integer quantization

This section presents a novel secure multiparty vector sum protocol for floating point real numbers based on a modified version of the ElGamal cryptosystem. This proposal is associated with Publication 3, Publication 5 and Publication 6.

The proposed protocol utilizes a modified version of the ElGamal encryption scheme to facilitate secure multiparty vector computation. This approach enables multiple clients to collaboratively calculate the sum of their encrypted vectors, while ensuring the confidentiality of their individual contributions from both the server and other participating clients. The protocol's security is founded upon the computational intractability of the discrete logarithm problem.

Due to the input requirement being floating-point real number vectors, the transformation and processing of these numbers to make them compatible with tradi-

Table 2.1: Notation and Explanation

Notation	**Explanation**
U_i	User i
U	User
S	Aggregation server
n	Number of participants
M	Number of training samples
W	Model parameter vector
V	Sum of n vectors
W^t	Global model parameter vector at round t
W_i^t	Model parameter vector of party i at round t
m_i	Number of data samples for user U_i
m	Total training data from all parties
$W_i^{(j)}$	Vector W of party i with elements indexed by j
x_i, y_i	Secret key vectors for ElGamal encryption of party i
X_i, Y_i	Public key vectors for ElGamal encryption of party i
c_i, d_i, p_i, q_i	Secret key vectors for ECC encryption of party i
C_i, D_i, P_i, Q_i	Corresponding public key vectors for ECC encryption of party i

tional cryptographic algorithms would demand extensive and inefficient computations due to their large size. Motivated by the natural resilience of deep neural networks to low-precision fixed-point representations [188], the thesis proposes an integer encoding method that rounds the client's model weights to reduce the cost during communication. The thesis employs a preliminary transformation of the input. The floating real numbers will be scaled and normalized to a suitable value range [0, 1]. After normalization, these real numbers will be multiplied by a coefficient known as precision, corresponding to the number of decimal places used to represent the number. This process assists in converting the numbers into appropriate integers for encryption operations. Optimal precision selection balances computational speed and efficiency, albeit at the cost of precision. Describe three steps of the procedure in detail as follows.

2.2.1. Proposed protocol

The proposed protocol consists of two phases illustrated as in Figure 2.1.

Input:

- Each party U_i has private vector $W_i = \{W_i^{(j)}, 1 \leq j \leq model\_size\}$.
- Each party U_i has two private key vectors: $x_i = \{x_i^{(j)}\}, y_i = \{y_i^{(j)}\}$.
- System parameters: the exponential factor (γ), $\mathbb{Z}_\mathbf{p}$ and generator g.

Output: Approximate vector sum: $\tilde{W} = \sum_{i=1}^{n} W_i$.

Phase 1: Initialization Phase

- Each party U_i sends its public key vectors $\{X_i^{(j)}\} = \{\mathbf{g}^{\mathbf{x}_i^{(j)}}\}, \{Y_i^{(j)}\} = \{\mathbf{g}^{\mathbf{y}_i^{(j)}}\}$, and normalization factor $(minW_i + \sigma_i, maxW_i + \sigma_i')$ to server.
- The server computes: $X = \left\{\prod_{i=1}^{n} X_i^{(j)}\right\}$; $Y = \left\{\prod_{i=1}^{n} Y_i^{(j)}\right\}$ for $1 \leq j \leq model\_size$ and $W_{max} = \max_{i=i}^{n}(maxW_i + \sigma_i')$ and $W_{min} = \min_{i=i}^{n}(minW_i + \sigma_i)$ then sends them back to all clients.

Phase 2: Main phase

- Each client quantize parameter vectors $\tilde{W}_i^{(j)} \leftarrow \frac{W_i^{(j)} - W_{min}}{W_{max} - W_{min}} 10^{\gamma}$, for $1 \leq j \leq model\_size$.
- Each party U_i encrypts his model's secret parameter vectors: $\left\{V_i^{(j)} = \frac{X^{(j)\mathbf{y}_i^{(j)}}}{Y^{(j)\mathbf{x}_i^{(j)}}} \mathbf{g}^{\tilde{W}_i^{(j)}}\right\}$ for $1 \leq j \leq model\_size$ and sends to the server.
- The server then computes $\{V^{(j)}\} = \left\{\prod_{i=1}^{n} V_i^{(j)}\right\}$ for $1 \leq j \leq model\_size$.
- The server performs Shank's algorithm to find $S^{(j)}$ with:

$$g^{S^{(j)}} = V^{(j)} \text{ for } 1 \leq j \leq model\_size.$$

- The server computes the vector sum by computes: $\frac{S^{(j)}}{10^{\gamma}}(W_{max} - W_{min}) + W_{min}$

Figure 2.1: Secure Vector Sum Protocol based on Integer quantization and Elgamal cryptosystem

Initialization phase

To initiate the protocol, the following parameters are required:

- In the initial step, before the protocol begins, the central aggregator selects a precision level, known as the exponential factor (γ), and broadcasts it to all clients. This factor determines the number of decimal places preserved in the quantization process.

- A prime p and another prime q are chosen such that $p-1$ is a multiple of q. Let $\mathbf{g}$ be a generator of the cyclic group $\mathbb{Z}_p$, with the conditions that $g \neq 1$ and $\mathbf{g}^q \mod p = 1$. All computations in the proposed protocol are carried out within $\mathbb{Z}_p$, and the parameters (p,q,g) are public, shared by the central server and all clients.

- Each client P_i possesses two private key vectors:

$$\mathbf{x}_i = \{x_i^{(1)},\dots,x_i^{(model\_size)}\}, \quad \mathbf{y}_i = \{y_i^{(1)},\dots,y_i^{(model\_size)}\},$$

where each $x_i^{(i)}, y_i^{(i)} \in \{1,2,\dots,p-1\}$, along with the corresponding public key vectors:

$$X_i = \{\mathbf{g}^{x_i^{(1)}},\dots,\mathbf{g}^{x_i^{(model\_size)}}\}, \quad Y_i = \{\mathbf{g}^{y_i^{(1)}},\dots,\mathbf{g}^{y_i^{(model\_size)}}\}.$$

Here, the private key $\mathbf{y}_i$ is used only once. Each client P_i submits the public keys $M_i = \{X_i, Y_i\}$ to the central server. In addition to the public key, the client also sends the normalization factors $(minW_i + \sigma_i, maxW_i + \sigma_i')$, with $\sigma_i < 0$ and $\sigma_i' > 0$, to the server. The server then precomputes:

$$X = \left\{\prod_{i=1}^{n} X_i^{(j)}\right\}, \quad Y = \left\{\prod_{i=1}^{n} Y_i^{(j)}\right\} \quad \text{for} \quad 1 \leq j \leq model\_size,$$

and computes $W_{max} = \max_{i=1}^{n}(maxW_i + \sigma_i')$ and $W_{min} = \min_{i=1}^{n}(minW_i + \sigma_i)$. These values, along with the computed public parameters $M = \{X, Y\}$ and W_{max}, W_{min}, are then broadcast to all clients through the public network.

Secure *n*-clients sum computation phase

After the necessary parameters have been initialized, the main phase of the secure sum protocol begins. This phase consists of two steps, which are outlined in detail below:

Step 1: Client Operations

Each client U_i performs the following tasks:

- First, the client applies several operations to convert floating-point parameters into integers. The private parameters are normalized to the range $[0,1]$, and then scaled by multiplying them by 10^γ, where γ is the exponential factor received from the central aggregator. Specifically, for each parameter $W_i^{(j)}$ at model dimension j, the client computes:

$$\tilde{W}_i^{(j)} \leftarrow \frac{W_i^{(j)} - W_{min}}{W_{max} - W_{min}} \cdot 10^\gamma, \quad \text{for} \quad 1 \leq j \leq \text{model\_size}.$$

- Next, the client encrypts its model's secret parameters using the public keys $M = \{X, Y\}$ and its private keys $\mathbf{x}_i$ and $\mathbf{y}_i$. The encrypted model parameters are encoded into a vector V_i as follows:

$$V_i = \left\{ V_i^{(j)} = \frac{X^{\mathbf{y}_i^{(j)}}}{Y^{\mathbf{x}_i^{(j)}}} \cdot \mathbf{g}^{\tilde{W}_i^{(j)}} \right\}, \quad \text{for} \quad 1 \leq j \leq \text{model\_size},$$

where $X^{\mathbf{y}_i^{(j)}}, Y^{\mathbf{x}_i^{(j)}}, \frac{X^{\mathbf{y}_i^{(j)}}}{Y^{\mathbf{x}_i^{(j)}}} \in \mathbb{Z}_p$. Once the vector V_i is computed, the client sends it to the central server.

Step 2: Server Operations

Upon receiving the encrypted vectors V_i from all clients, the server performs the following operations:

- The server computes the aggregated vector V by multiplying the corresponding components of all received V_i's:

$$V^{(j)} = \prod_{i=1}^{n} V_i^{(j)}, \quad \text{for} \quad 1 \leq j \leq \text{model\_size}.$$

- The server then applies Shanks' algorithm to compute the vector $S = \{S^{(j)}\}$, where each component $S^{(j)}$ satisfies the equation:

$$g^{S^{(j)}} = V^{(j)}, \quad \text{for} \quad 1 \leq j \leq \text{model\_size}.$$

Since the values of $S^{(j)}$ are not large, solving the discrete logarithm problem is computationally feasible.

- Finally, the server computes the vector sum by reversing the scaling process and applying the normalization:

$$\frac{S^{(j)}}{10^\gamma} \cdot (W_{max} - W_{min}) + W_{min}, \quad \text{for} \quad 1 \leq j \leq \text{model\_size}.$$

2.2.2. Estimation error evaluation

In this section, instead of directly computing the exact sum of the vectors, the protocol employs integer quantization, which introduces rounding errors into the final aggregated sum. The following theorem provides an in-depth analysis of the relative error involved and establishes the maximum allowable threshold for this error during the computation.

Theorem 2.2.1. *The protocol described in Figure 2.1 can approximate the sum of n vectors, with the error bound for each j-th component given by the formula:*

$$\Delta S^{(j)} = \sqrt{(\delta_1^{(j)})^2 + (\delta_2^{(j)})^2 + \ldots + (\delta_n^{(j)})^2} \leq \varepsilon_d(n+1),$$

where ε_d denotes the error incurred when using d decimal digits for rounding, $\Delta S^{(j)}$ represents the total relative error for the j-th component of the summed vector, and $\delta_i^{(j)}$ is the relative error of the j-th component of the local vector after rounding.

Proof. Since the components of the vector are computed independently, we can, without loss of generality, prove that if the central server calculates a value $S^{(j)}$ such that the equation $\mathbf{g}^{S^{(j)}} = V^{(j)}$ holds, then $S^{(j)}$ is an estimate of the sum of all clients' secret values. To simplify the exposition, we omit the index (j) in the following discussion, though it is understood that the proof applies to the j-th component of the vector.

Assume that $\mathbf{g}^S = V$. We can then express V as:

$$\begin{aligned}\mathbf{g}^S = V = \prod_{i=1}^{n} V_i = \prod_{i=1}^{n} \frac{\mathbf{g}^{\tilde{W}_i} X^{\mathbf{y}_i}}{Y^{\mathbf{x}_i}} &= \mathbf{g}^{\sum_{i=1}^{n} \tilde{W}_i} \prod_{i=1}^{n} \frac{(\prod_{k=1}^{n} X_k)^{\mathbf{y}_i}}{(\prod_{k=1}^{n} Y_k)^{\mathbf{x}_i}} \\ = \mathbf{g}^{\sum_{i=1}^{n} \tilde{W}_i} \prod_{i=1}^{n} \frac{\left(\mathbf{g}^{\sum_{j=1}^{n} \mathbf{x}_j}\right)^{\mathbf{y}_i}}{\left(\mathbf{g}^{\sum_{j=1}^{n} \mathbf{y}_j}\right)^{\mathbf{x}_i}} &= \mathbf{g}^{\sum_{i=1}^{n} \tilde{W}_i} \frac{\mathbf{g}^{\sum_{j=1}^{n} \mathbf{x}_j \sum_{i=1}^{n} \mathbf{y}_i}}{\mathbf{g}^{\sum_{j=1}^{n} \mathbf{y}_j \sum_{i=1}^{n} \mathbf{x}_i}} = \mathbf{g}^{\sum_{i=1}^{n} \tilde{W}_i}.\end{aligned}$$

Let $S^{(j)} = \sum_{i=1}^{n} \tilde{W}_i^{(j)}$, meaning $S = \{S^{(j)}\}$ represents the approximate sum of the vectors after the rounded private vectors have been summed.

By the definition of relative error for rounding, let $\delta_i^{(j)}$ represent the relative error in rounding the j-th component from client U_i. Consequently, the relative error in summing these components is given by:

$$\Delta S^{(j)} = \sqrt{(\delta_1^{(j)})^2 + (\delta_2^{(j)})^2 + \ldots + (\delta_n^{(j)})^2}.$$

Following the rounding error threshold rule, we have:

$$\Delta S^{(j)} \leq \varepsilon_d(n+1).$$

□

2.2.3. Privacy analysis

In this section of the dissertation, we will provide a thorough proof demonstrating that the multi-party secure summation protocol we have introduced guarantees the privacy of each semi-honest client. This is in accordance with the following theorem:

Theorem 2.2.2. *The secure summation protocol for n clients, as presented in Figure 2.1, ensures that the privacy of each honest client is protected against the server and up to $(n-2)$ corrupted clients in a semi-honest model.*

Proof. Similarly to the aforementioned section, due to the independent computation of vector components and the use of distinct keys, we can, without loss of generality, demonstrate the security of the protocol for any component j. Firstly, the thesis proves that the multi-party secure summation protocol ensures the privacy of each participant's input in the semi-honest model without colluding among the participants, under the DDH assumptions.

During phase 1, party U_i transmits $X_i^{(j)} = g^{x_i^{(j)}}$ and $Y_i^{(j)} = g^{y_i^{(j)}}$, where $x_i^{(j)}$ and $y_i^{(j)}$ are randomly selected values, uniformly distributed over $[1, p-1]$. Therefore, the probability of choosing any given value is equal and is $\frac{1}{p}$. Consequently, $X_i^{(j)}$ and $Y_i^{(j)}$ are also random variables in accordance with the ElGamal encryption algorithm. As per the Diffie-Hellman decisional assumption:

$$\left|\Pr\left[A\left(\mathbb{G}, q, g, g^{x_i^{(j)}}, g^{y_i^{(j)}}\right) = 1\right] - \Pr\left[A\left(\mathbb{G}, q, g, x_i^{(j)}, y_i^{(j)}\right) = 1\right]\right| < \mu(n) = \left|\frac{1}{p}\right|.$$

Similarly, σ_i and σ_i' are randomly chosen numbers with a uniform distribution, thus $\min W_i + \sigma_i$ and $\max W_i + \sigma_i'$ also have a uniform distribution. Therefore, both of these sequences are computationally indistinguishable.

At step 2, $V_i = \left\{ V_i^{(j)} = \frac{X^{\mathbf{y}_i^{(j)}}}{Y^{\mathbf{x}_i^{(j)}}} \mathbf{g}^{\tilde{W}_i^{(j)}} \right\}$ is transmitted, but as proven above with

$\mathbf{x}_i^{(j)}$ and $\mathbf{y}_i^{(j)}$ being randomly chosen values with a uniform distribution, $V_i^{(j)}$ also has a uniform distribution and is thus also computationally indistinguishable:

$$\left|\Pr\left[A\left(\mathbb{G},q,g,g^{x_i^{(j)}},g^{y_i^{(j)}},V_i^{(j)},\gamma\right)=1\right]-\Pr\left[A\left(\mathbb{G},q,g,x_i^{(j)},y_i^{(j)},\tilde{W}_i^{(j)},\sigma_i,\sigma_i'\right)=1\right]\right|<\mu(n).$$

Hence, the proposed protocol guarantees the privacy of each participant's input in the actual model.

Subsequently, the thesis establishes that the multi-party secure sum protocol preserves the privacy of honest users, even in the case where $n-2$ out of n users, who are assumed to be semi-honest, collude.

To prove that the protocol protects the privacy of the honest parties against collusion by up to $n-2$ semi-honest parties and the computation party, one must present a simulator M that can simulate what the dishonest parties and the computation party can observe during the protocol execution using a polynomial-time algorithm. Specifically, it is necessary to exhibit a polynomial-time algorithm that computes the joint view of the computation party and the dishonest parties using only the knowledge of the semi-honest parties, the outputs, the public keys, and a number of ElGamal ciphertexts on the prime field $\mathbb{Z}_p$.

Without the loss of generality, we assume that two clients U_1 and U_2 do not collude while the central server and the others $U_i \| i \in I = \{3,4,\ldots,n\}$ collude with each other. In the secure protocol, each client only sends the encrypted value $V_i^{(j)}$ and two public keys $X_i^{(j)},Y_i^{(j)}$ to the server. $X_i^{(j)},Y_i^{(j)}$ are random values because the private keys $x_i^{(j)},y_i^{(j)}$ are uniformly random. To prove the theorem, we must construct a probabilistic polynomial-time algorithm that can simulate the computation for the messages $\tilde{W}_1$ and $\tilde{W}_2$ using only the final sum $S^{(j)}$, corrupted clients' knowledge $\{x_i^{(j)},y_i^{(j)},V_i^{(j)}\}$ and public keys $X_1^{(j)},Y_1^{(j)},X_2^{(j)},Y_2^{(j)}$.

We denote the algorithm that satisfies the above assumption as M. Algorithm M uses $(u_{12},v_{12})=(g^{\tilde{W}_1^{(j)}}g^{x_2^{(j)}y_1^{(j)}},g^{x_2^{(j)}})$, $(u_{21},v_{21})=(g^{\tilde{W}_2^{(j)}}g^{x_1^{(j)}y_2^{(j)}},g^{x_1^{(j)}})$ as its input to simulate $\tilde{W}_1^{(j)}, \tilde{W}_2^{(j)}$ as follows:

$$U_1^{(j)} = \frac{u_{12}.Y_1^{\sum_{i\in I} x_i^{(j)}}.g^{S^{(j)}-\sum_{i\in I}\tilde{W}_i^{(j)}}}{u_{21}.X_1^{\sum_{i\in I} y_i^{(j)}}}$$
$$U_2^{(j)} = \frac{u_{21}.Y_2^{\sum_{i\in I} x_i^{(j)}}.g^{S^{(j)}-\sum_{i\in I}\tilde{W}_i^{(j)}}}{u_{12}.X_2^{\sum_{i\in I} y_i^{(j)}}}$$

And so:

$$\left\{M(I,\bar{x}_I^{(j)}, f_I(\bar{x}^{(j)}))\right\} = \left\{\left[X_i^{(j)}, Y_i^{(j)}\right]_{i=1}^n, X^{(j)}, Y^{(j)}, \left[V_i^{(j)}\right]_{i=1}^n, S^{(j)}, V^{(j)}, U_1^{(j)}, U_2^{(j)}\right\}. \tag{2.2.1}$$

Whereas:

$$\left\{VIEW_{A,I}^{\pi}(\bar{x}^{(j)})\right\} = \left\{\left[X_i^{(j)}, Y_i^{(j)}\right]_{i=1}^n, X^{(j)}, Y^{(j)}, \left[V_i^{(j)}\right]_{i=1}^n, S^{(j)}, V^{(j)}\right\}. \tag{2.2.2}$$

Therefore, we can see that: $\left\{VIEW_{A,I}^{\pi}(\bar{x}^{(j)})\right\}$ and $\left\{M(I,\bar{x}_I^{(j)}, f_I(\bar{x}^{(j)}))\right\}$ differ only in their $U_1^{(j)}, U_2^{(j)}$ values.

Given that $x_1^{(j)}, x_2^{(j)}, y_1^{(j)}, y_2^{(j)}$ are randomly selected from $\mathbb{Z}_p^*$, under the Decisional Diffie-Hellman (DDH) assumption, the elements $u_{12}, u_{21}, v_{12}, v_{21}$ along with $x_1^{(j)}, x_2^{(j)}, y_1^{(j)}, y_2^{(j)}$ are computationally indistinguishable.

According to the definition, U_1 and U_2 are dependent random variables determined by u_{12}, u_{21}, v_{12}, and v_{21}. These variables, in turn, depend on the independent random variables $x_1^{(j)}$, $x_2^{(j)}$, $y_1^{(j)}$, and $y_2^{(j)}$, which are uniformly distributed as they are randomly selected from the set $(1, p-1)$. Consequently, U_1 and U_2 are also uniformly distributed and thus thay and $x_1^{(j)}, x_2^{(j)}, y_1^{(j)}, y_2^{(j)}$ are indistinguishable. This implies that even with additional knowledge about U_1 and U_2, corrupted parties and the aggregation server gain no further information beyond the publicly shared parameters and the data derived from collusion.

According to Definition 1.3.4, the protocol is *semantically secure* against the collusion of the adversarial party and up to $n-2$ semi-honest parties.

On the other hand, the parameters shared among the parties are based on the discrete logarithm problem over the prime field $\mathbb{Z}_p$. This problem currently belongs to the NP-hard class; therefore, reconstructing the secret parameters from the publicly shared parameters is also an NP-hard problem. Given the current computational capabilities, when the secret parameters are randomly chosen from $(1, p-1)$ in accordance with safety conditions, the protocol also ensures *computational security* based on the difficulty of the discrete logarithm problem in the finite prime field $\mathbb{Z}_p$. $\square$

2.2.4. Performance Evaluation

2.2.4.1. Computational cost

In this section, the thesis mainly focuses on the time complexity. To facilitate the discussion, the following notations in Table 2.2 will be used:

Table 2.2: Computational complexity notations

Notation	Time required to perform
T_E	A exponentiation operation on $\mathbb{Z}_p$.
T_M	A multiplication operation on $\mathbb{Z}_p$.
T_I	An inversion operation on $\mathbb{Z}_p$.
T_S	The Shanks' baby-step giant-step algorithm.

We analyze the time complexity the proposed protocol by each phase:

- Initialization phase:

  Each client. In the system preparation stage, each client computes public keys by performing exponentiation operation on $\mathbb{Z}_p$. This performance takes each client $model\_size \times T_E$ for each key. Since each client possesses two public keys, the time taken by each client to prepare both public keys is $2 \times model\_size \times T_E$.

  Server. The server calculates the public parameters X, and Y by multiplying the corresponding public keys of all clients, where each public key corresponds to a matrix of points. As a result, the time taken by the server in this phase can be expressed as $2 \times n \times model\_size \times T_M$.

- Main phase:

  Each client. Each client generates V_i with $2 \times model\_size$ multiplication operations $model\_size$ inversion operations and $3 \times model\_size$ exponentiation operations. So, this cost: $2 \times model\_size \times T_M + model\_size \times T_I + 3 \times model\_size \times T_E$

  Server. For server side, server first consumes $model\_size T_M$ for computing V. Server then takes $model\_size \times T_S$ for finding s that satisfy $g^s = V^j$. Finally, server consumes $model\_size(T_M + T_S)$ since performing operations among integer matrices.

In the calculations mentioned, the exponentiation of two numbers over Z_p requires the highest computational cost, significantly more than both the inversion and multiplication of elements in Z_p. The execution time for Shank's algorithm is also directly derived from the run-time of the exponentiation algorithm and requires performing many exponentiations to achieve the desired value. Therefore, for the client, the encryption time will approximate the time for calculating exponentiations, while for the server, the decryption time will approximate the time it takes to execute Shank's algorithm.

To investigate the impact of decimal precision on encryption time, we conducted an experiment using identical computer configurations across all cases. This involved a single-threaded Python client running on an i7 processor without GPU support, and we excluded communication latency from our measurements. The results are summarized in Figure 2.2. The amount of time needed to compute the encryption of a particular parameter is measured and correlated to the model size.

Our findings demonstrate a clear trend: encryption time decreases as the decimal precision of the model parameters is reduced, while the size of the private vectors remains constant. This provides strong evidence that our proposed compression technique effectively enhances encryption efficiency. Furthermore, for each level of decimal precision, encryption time increases as the size of the private vectors in the

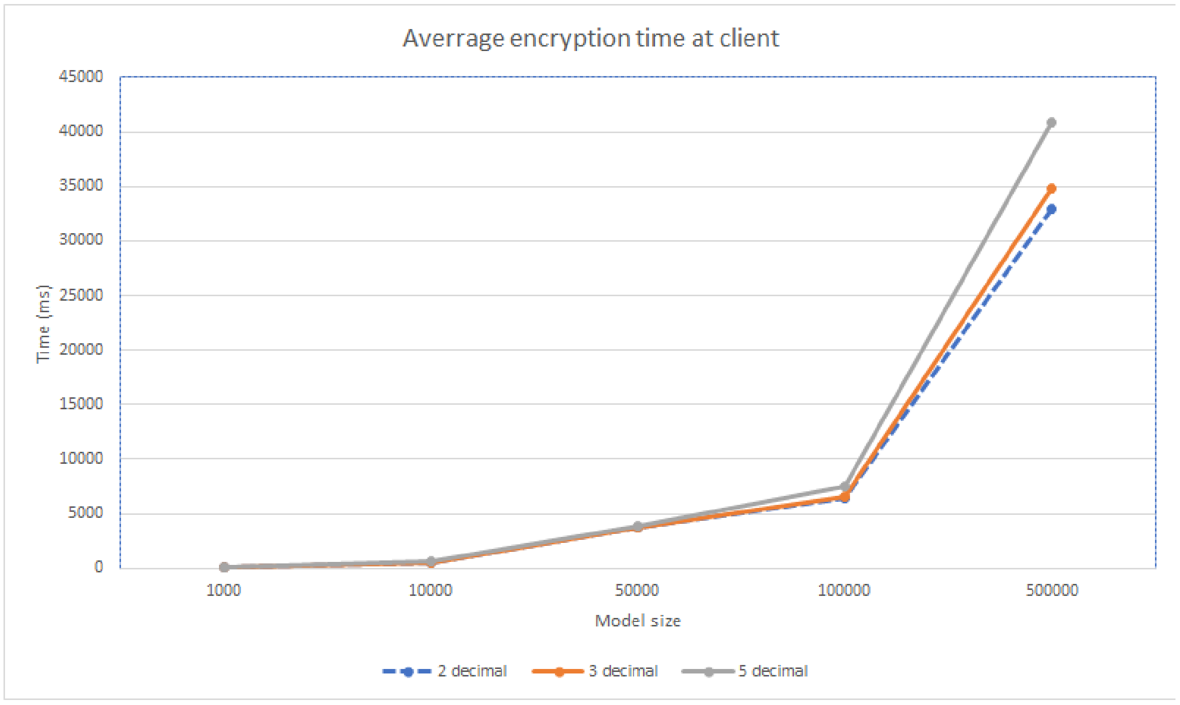


Figure 2.2: The average client encryption time at decimal precision levels of 2, 3, and 5

framework grows, likely due to the greater computational demands associated with larger private vectors. It is evident that the encryption time at the client side remains consistent across different nodes and is independent of the number of participating nodes. This is more efficient than the secret-sharing-based protocols proposed in works such as [162], where the computational complexity at the client side depends on the number of participating nodes. Consequently, these protocols often struggle to scale effectively in models with many participants.

The observed reduction in encryption time has a significant impact on the scalability of deployments, particularly for large federated learning networks where computational efficiency is paramount. By reducing encryption time, our approach can enable the implementation of FL in scenarios where computational resources might be limited.

The sum computational time is the time taken to execute Shank's algorithm on the obtained results. The complexity of this algorithm is exponential, therefore the execution is relatively slow for large numbers. The decryption execution time on the server is presented in Figure 2.3.

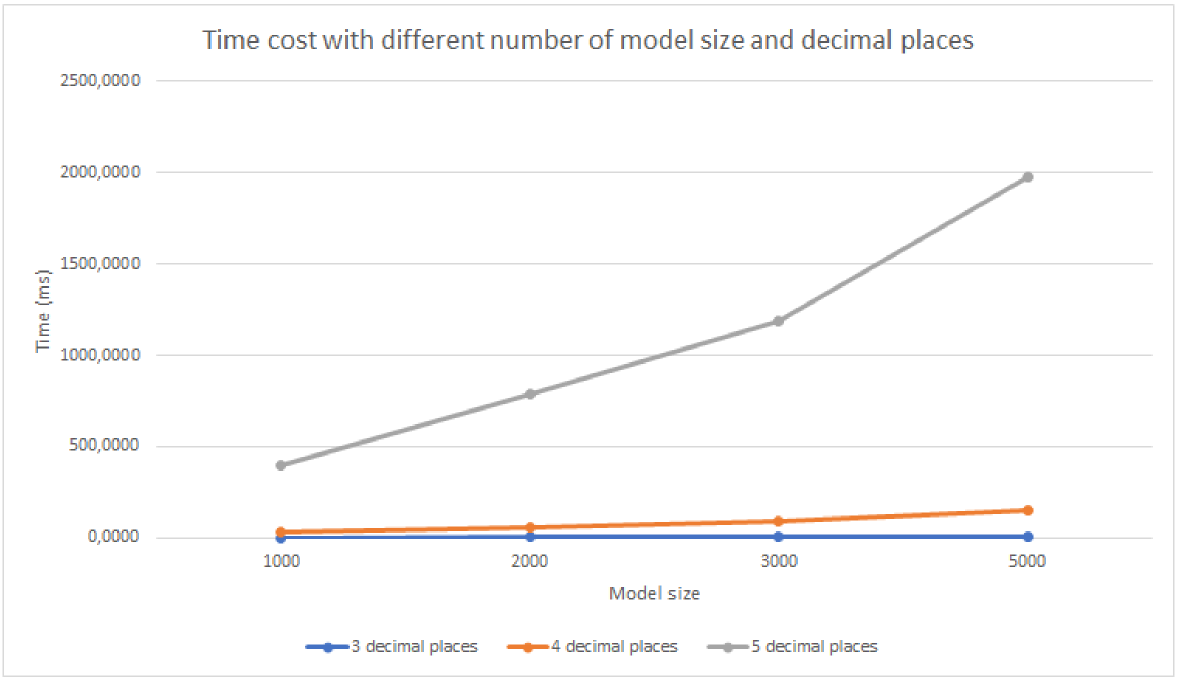


Figure 2.3: The computational time required for the decryption of different model size at decimal precision levels of 3, 4, and 5

The results clearly demonstrate that as the required precision level increases, the time needed to compute the sum at the server rises significantly, following an exponential trajectory. This finding highlights the intrinsic computational challenges associated with higher precision, as it necessitates a substantial enhancement in the server's processing capabilities. Furthermore, while the growth in the number of participating parties naturally leads to a larger aggregate sum and consequently an increase in the server's computation time, this relationship is linear rather than exponential. This linear increase, characterized by a simple expansion in the volume of values to be processed, remains well within manageable limits, ensuring the system's scalability and efficiency.

2.2.4.2. Communication cost

We denote the size in bits of a private vector (W_i) as size_m and the size of public keys in Elgamal protocols as size_k.

In the initialization phase, each client needs to send its public keys to the server. This process requires a total bandwidth of $2 \times model\_size \times \text{size}_k$ bits. The server then broadcasts X, Y to every client, which requires the server to establish n connections, each with a bandwidth of $2 \times model\_size \times \text{size}_k$ bits. Therefore, the total bandwidth for the initialization phase is $4 \times n \times model\_size \times \text{size}_k$ bits.

Moving on to the main phase, each client is required to send V_i to the server. The transmission of V_i messages requires $model\_size \times \text{size}_k$ bits of bandwidth for each message. In the second computation phase, sever needs to send the sum S all clients, which requires a bandwidth of size_m bits for each message. The total bandwidth for this phase is $n \times \text{size}_m$ bits.

For instance, with a private vector of 50,000 dimensions and an Elgamal encryption key of 256 bits, the total communication bandwidth during each round of the protocol is summarized in Table 2.3.

Table 2.3: Communication bandwidth of the proposed quantization-based protocol with model size is 50,000 and Elgamal key size if 256 bit

	Client i	**Server**
Initialization Phase	3.07 MB	3.07 MB $\times n$
Computation Phase	1.53 MB	0.38 MB $\times n$

While it is evident that the overall communication cost has increased significantly compared to the original federated learning framework, this added expense is justified when considering the enhanced privacy and security guarantees provided by the protocol. Importantly, despite the rise in communication overhead, the total cost remains far more efficient than what is typically incurred in secret-sharing-based protocols, such as those discussed in [162]. These protocols, while secure, often require a much greater volume of data exchange between parties, leading to substantially higher communication costs. In contrast, the approach presented here achieves a more balanced trade-off between security and efficiency, ensuring robust protection of sensitive data without imposing prohibitive communication overheads.

2.2.5. Discussion

Building on the analysis presented, it becomes clear that the first proposed protocol is particularly effective when dealing with lower precision requirements. As the precision level increases, the computational demands on the server side escalate significantly, reflecting the exponential nature of the computational burden associated with higher precision. In contrast, the client-side costs remain relatively stable, showing only a linear increase with the size of the private vector.

Communication costs for clients are fixed and primarily determined by the size of the private vector and the encryption key used. Meanwhile, the server's bandwidth requirements are influenced by both the precision of the numerical representations and the number of participants involved. Although the precision of the vector summation may decrease due to the compression technique, the impact on the overall accuracy of the federated learning model is minimal. In fact, in some instances, this slight loss in precision may contribute positively by introducing additional randomization during the training process, which can enhance the model's ability to generalize. A more detailed evaluation of these factors and their implications on federated learning will be thoroughly explored in Chapter 3.

To address the challenges of declining accuracy in computations and the significant computational costs required for high precision, this dissertation presents a second protocol designed for securely computing the summation of multiple private real number vectors. This protocol leverages a masking matrix technique and a modified ECC encryption scheme. A comprehensive discussion of this protocol is provided in the following section of the dissertation.

2.3. Secure multi-party sum protocol using mask matrix with Modified ECC protocol

To develop the second new SMC protocol, it is crucial to address the limitations of existing approaches. The first protocol faces a significant drawback: its computational cost increases substantially with the required accuracy. This is largely due to the time-intensive nature of implementing Shank's algorithm, especially when dealing with large integers correlated with high decimal precision. While separating a number into its real and integer parts can potentially simplify computations, this method inadvertently exposes the decimal components, compromising privacy. To overcome this challenge, a more effective solution involves multiplying the decimal parts by a masking matrix, thereby concealing them and ensuring privacy. This approach not only reduces computational costs but also enhances the security of the protocol, making it a more robust solution for secure computations involving real number vectors.

In this section, the dissertation introduces the second proposed protocol, which utilizes a masking matrix technique combined with a variant of the ECC cryptosystem. In this protocol, the real part of the vector is concealed using the masking matrix, while the integer part is secured with ECC encryption. During the aggregation process on the server side, the masking matrices cancel out, allowing the desired sum to be obtained.

The ECC scheme is a public-key cryptography system that leverages the properties of elliptic curves. In the ECC system, encryption and decryption are performed using elliptic curve point operations rather than the modular arithmetic operations used in the traditional ElGamal system. This approach enhances the efficiency of ECC compared to the traditional Elgamal system, as elliptic curve point operations are significantly faster than modular arithmetic operations. The security of ECC relies on the complexity of the Elliptic Curve Discrete Logarithm Problem (ECDLP), which involves determining the private key associated with a given public key. This complexity makes ECC not only more efficient but also more secure than the traditional ElGamal system, which has led to its widespread use in modern cryptography.

It is worth noting that the ECC encryption system can be replaced with the traditional ElGamal encryption system. However, this substitution would require altering the operations from elliptic curve point multiplication to modular exponentiation and would necessitate longer key lengths to achieve the same level of security.

2.3.1. Proposed Protocol

We propose a SMC protocol that leverages the additively homomorphic property, which enables multiple parties to collaboratively compute the sum of their private messages without disclosing the actual values to one another. This protocol utilizes an elliptic curve analog of the ElGamal cryptographic system, enhancing security and efficiency. The implementation of our protocol necessitates a preparatory stage where all necessary parameters are established. It is structured into three distinct phases: an initial setup phase, followed by two computational phases, where the participants engage in secure computations. Figure 2.4 provides a comprehensive overview of the protocol, illustrating its structure and the flow between different phases.

Initialization phase

- All parties choose an elliptic curve $E(\mathbb{Z}_{\mathbf{q}})$ with a point O at infinity and $\mathbf{q}$ to be a large prime, in which the elliptic curve discrete logarithm problem is hard. In addition, G is a generator point of the elliptic curve E with order $\mathbf{q}$ (i.e., $\mathbf{q} \cdot G = 0$). The curve $E(\mathbb{Z}_{\mathbf{q}})$ and the generator point G are public to the server and all clients.
- Every client possesses a confidential vector W_i in the form of a vector comprising floating-point real numbers. Let W_i be a vector of dimension *model_size*. We transform this vector into a matrix W of size $d \times d$, where $d = \lceil \sqrt{model\_size} \rceil$ is the ceiling of the square root of *model_size*. The elements of the vector W_i are arranged sequentially across the rows to form the corresponding matrix W. Formally, if $W_i = [w_1, w_2, \ldots, w_{model\_size}]$, then the matrix W can be expressed as:

$$W = \begin{pmatrix} w_1 & w_2 & \cdots & w_d \\ w_{d+1} & w_{d+2} & \cdots & w_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ w_{(d-1)d+1} & w_{(d-1)d+2} & \cdots & w_{model\_size} \end{pmatrix}$$

Input:

- Each party U_i has private matrix $\overline{W}_i = [W_i^{(kj)}]; 1 \leq j,k \leq d$.
- Each party U_i has four private key matrices: $p_i = [p_i^{(kj)}], q_i = [q_i^{(kj)}], c_i = [c_i^{(kj)}], d_i = [d_i^{(kj)}]$.
- Each party U_i has four private random matrices: M_i, N_i, r_i, s_i.
- System parameters: Elliptic Curve $E(\mathbb{Z}_\mathbf{q})$ with order $\mathbf{q}$ and generator point G.

Output: Sum vector: $W = \sum_{i=1}^{n} W_i$.

Phase 1: Initialization Phase

- Set up system parameters $E(\mathbb{Z}_\mathbf{q})$ and generator point G.
- Each party U_i sends its public keys $P_i = \{p_i^{(kj)}G\}, Q_i = \{q_i^{(kj)}G\}$, and $C_i = \{c_i^{(kj)}G\}, D_i = \{d_i^{(kj)}G\}$ to server.
- The server computes and broadcasts: $P = \sum_{i=1}^{n} P_i, Q = \sum_{i=1}^{n} Q_i, C = \sum_{i=1}^{n} C_i, D = \sum_{i=1}^{n} D_i$.

Phase 2: Main phase

- Each party U_i computes and send his model's public parameter vectors to the server:
  $A_i = M_i + r_i, B_i = N_i + s_i$,
  $R_i = \{r_i^{(kj)}G + q_i^{(kj)}P^{(kj)} - p_i^{(kj)}Q^{(kj)}\}, S_i = \{s_i^{(kj)}G + c_i^{(kj)}D^{(kj)} - d_i^{(kj)}C^{(kj)}\}$
- The server then computes $R = \sum_{i=1}^{n} R_i, S = \sum_{i=1}^{n} S_i$ and find r and s that each element satisfy $r^{(kj)}G = R^{(kj)}$ and $s^{(kj)}G = S^{(kj)}$ and send $M = \sum_{i=1}^{n} A_i - r, N = \sum_{i=1}^{n} B_i - s$ to all clients
- Each party computes $T_i = W_i + M_i N - M N_i$ and sends T_i to the server
- The server obtain the sum of all clients' messages as $T = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} W_i = W$.

Figure 2.4: Secure Vector Sum Protocol based on Mask matrix combine with ECC cryptosystem

If *model_size* is not a perfect square, the remaining elements of the matrix W are padded as necessary, typically with zeros, to fill the matrix completely.

- Each client U_i has already chooses four private keys p_i, q_i, c_i, d_i where each private key is a matrix of dimensions $d \times d$ with each element in the range $[1, q-1]$. Client U_i then computes the corresponding public keys:

$$P_i = \{P_i^{(kj)}\}_{d\times d} = \{p_i^{(kj)} \cdot G\}_{d\times d},$$
$$Q_i = \{Q_i^{(kj)}\}_{d\times d} = \{q_i^{(kj)} \cdot G\}_{d\times d},$$
$$C_i = \{C_i^{(kj)}\}_{d\times d} = \{c_i^{(kj)} \cdot G\}_{d\times d},$$
$$D_i = \{D_i^{(kj)}\}_{d\times d} = \{d_i^{(kj)} \cdot G\}_{d\times d}.$$

Each client then sends public keys P_i, Q_i, C_i, D_i to the central server.

- Receiving P_i, Q_i, C_i, D_i $(i = 1, \ldots, n)$, the server compute the public parameters $P = \sum_{i=1}^{n} P_i$, $Q = \sum_{i=1}^{n} Q_i$, $C = \sum_{i=1}^{n} C_i$, $D = \sum_{i=1}^{n} D_i$ and then broadcast P, Q, C, D to every client.

Secure sum computation phase

- Each client U_i chooses four private matrices M_i, N_i, r_i, s_i of size $d \times d$ and computes the flowing messages

$$A_i = M_i + r_i$$
$$B_i = N_i + s_i$$
$$R_i = \{R_i^{(kj)}\}_{d\times d} = \{r_i^{(kj)}G + q_i^{(kj)}P^{(kj)} - p_i^{(kj)}Q^{(kj)}\}_{d\times d}$$
$$S_i = \{S_i^{(kj)}\}_{d\times d} = \{s_i^{(kj)}G + c_i^{(kj)}D^{(kj)} - d_i^{(kj)}C^{(kj)}\}_{d\times d}$$

  and sends these computed messages $\{A_i, B_i, R_i, S_i\}$ to the server.

- At the second step, the central server computes $R = \sum_{i=1}^{n} R_i$ and $S = \sum_{i=1}^{n} S_i$. The server then finds r and s that each element $r^{(kj)}$ and $s^{(kj)}$ satisfy $r^{(kj)}G = R^{(kj)}$ and $s^{(kj)}G = S^{(kj)}$. From r and s, the central server obtains

$$M = \sum_{i=1}^{n} A_i - r = \sum_{i=1}^{n} M_i,$$
$$N = \sum_{i=1}^{n} B_i - s = \sum_{i=1}^{n} N_i$$

  Finally, the central server broadcasts M and N to all clients.

- Each client computes $T_i = W_i + M_i N - M N_i$ and sends T_i to the server.

- The central server obtains the sum of all clients' private messages as the flowing formula:

$$T = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} W_i = W. \tag{2.3.3}$$

2.3.2. Proof of correctness

Theorem 2.3.1. *The proposed protocol in the Figure 2.4 can calculate the sum of n vectors.*

Proof. We have:

$$R = \sum_{i=1}^{n} R_i = \left\{ \sum_{i=1}^{n} (r_i^{(kj)} G + q_i^{(kj)} P^{(kj)} - p_i^{(kj)} Q^{(kj)}) \right\}_{d\times d}.$$

Using the distributive property of scalar multiplication, we can write this as:

$$\begin{aligned} R = \{rG\}_{d\times d} &= \{(r_1^{(kj)} + r_2^{(kj)} + \cdots + r_n^{(kj)})G + (q_1^{(kj)} + q_2^{(kj)} + \cdots + q_n^{(kj)})P \\ &\quad - (p_1^{(kj)} + p_2^{(kj)} + \cdots + p_n^{(kj)})Q\}_{d\times d} \\ &= \{(r_1^{(kj)} + r_2^{(kj)} + \cdots + r_n^{(kj)})G + Q^{(kj)}P^{(kj)} - P^{(kj)}Q^{(kj)}\}_{d\times d} \\ &= \{(r_1^{(kj)} + r_2^{(kj)} + \cdots + r_n^{(kj)})G\}_{d\times d} \end{aligned}$$

Therefore, we have:

$$r^{(kj)} = r_1^{(kj)} + r_2^{(kj)} + \cdots + r_n^{(kj)} \text{for all } 1 \le k, j \le d.$$

In another hand, we have:

$$M = \sum_{i=1}^{n} A_i - r = \sum_{i=1}^{n} (M_i + r_i) - r = \sum_{i=1}^{n} M_i + \sum_{i=1}^{n} r_i - r$$

Thus

$$M = \sum_{i=1}^{n} M_i.$$

The proof of correctness is similar with

$$N = \sum_{i=1}^{n} N_i.$$

Using the formula for T_i, we have:

$$T_i = W_i + M_iN - MN_i$$

Summing over all T_i:

$$\begin{aligned}\sum_{i=1}^{n} T_i = \sum_{i=1}^{n}(W_i + M_iN - MN_i) &= \sum_{i=1}^{n} W_i + \sum_{i=1}^{n} M_iN - M\sum_{i=1}^{n} N_i \\ &= \sum_{i=1}^{n} W_i + MN - MN \\ &= \sum_{i=1}^{n} W_i\end{aligned}$$

Therefore, we have shown that $\sum\limits_{i=1}^{n} T_i$ is equal to $\sum\limits_{i=1}^{n} W_i$, which implies that $T = \sum\limits_{i=1}^{n} W_i$. □

2.3.3. Privacy analysis

In this section, the thesis aims to analyze the security of the proposed protocol under the assumption that all participating clients are semi-honest and fully comply with the protocol, along with all of the previously mentioned assumptions.

Theorem 2.3.2. *The protocol for secure n-clients sum presented in Figure 2.4 protects each honest client' privacy against the server and up to* $(n-2)$ *corrupted clients (and colluding with the server) in semi-honest model.*

Proof. Firstly, the thesis demonstrates the *computational security* of the proposed protocol by relying on the difficulty of the elliptic curve discrete logarithm problem.

During the initialization phase, each client utilizes their private keys to generate the corresponding public keys, which are then sent to the central server. An attacker could theoretically deduce the private keys p_i, q_i, c_i, d_i from the public keys P_i, Q_i, C_i, D_i if they manage to solve the elliptic curve discrete logarithm problem (ECDLP). However, the proposed protocol is based on the premise that the ECDLP is challenging to solve, which implies that an attacker cannot efficiently recover private keys from public keys. Therefore, the protocol maintains computational security against any semi-honest client attempting to discover other clients' private keys during this phase.

In the second phase, user U_i sends the values A_i, B_i, R_i, and S_i to the server. The component $R_i^{(kj)}$ includes two elements randomly distributed over $q_i^{(kj)} P^{(kj)}$ and $p_i^{(kj)} Q^{(kj)}$, and an additional value derived from point multiplication on an elliptic curve. Given the computational challenges posed by the elliptic curve discrete logarithm problem (ECDLP), it is infeasible to determine $r_i^{(kj)}$, $q_i^{(kj)}$, and $p_i^{(kj)}$ from R_i, as well as $s_i^{(kj)}$, $c_i^{(kj)}$, and $d_i^{(kj)}$ from S_i for all $1 \leq k, j \leq d$. Consequently, since $r_i^{(kj)}$ and $s_i^{(kj)}$ cannot be determined within a polynomially feasible time frame, neither N_i nor M_i can be derived from A_i and B_i. This phase of the protocol is thus computationally secure. Similarly, the variables M_i, N_i, and W_i derived from T_i remain secure, as they are random variables uniformly distributed, further enhancing the protocol's security.

Subsequently, the thesis proves that the multi-party secure summation protocol ensures the privacy of each participant's input in the semi-honest model without colluding among the participants, under the Decisional Diffie-Hellman (DDH) assumption. Due to the independent computation of vector components and the use of distinct keys, we can, without loss of generality, demonstrate the security of the protocol for any component (kj) with all $1 \leq k, j \leq d$.

During phase 1, party U_i transmits $P_i^{(kj)} = p_i^{(kj)} G$, $Q_i^{(kj)} = q_i^{(kj)} G$, $C_i^{(kj)} = c_i^{(kj)} G$, and $D_i^{(kj)} = d_i^{(kj)} G$ where $p_i^{(kj)}$, $q_i^{(kj)}$, $c_i^{(kj)}$, and $d_i^{(kj)}$ are randomly selected independent values, uniformly distributed over $[1, \mathbf{q}-1]$. Therefore, the probability of choosing any given value is equal and is $\frac{1}{\mathbf{q}}$. Consequently, $P_i^{(kj)}$, $Q_i^{(kj)}$, $C_i^{(kj)}$, and $D_i^{(kj)}$ are also random variables in accordance with the ECC encryption algorithm. As per the ECDLP assumption:

$$|\Pr[A(E(\mathbb{Z}_\mathbf{q}), P_i, Q_i, C_i, D_i) = 1] - \Pr[A(E(\mathbb{Z}_\mathbf{q}), p_i, q_i, c_i, d_i) = 1]| < \mu(n)$$

with $\mu(n)$ being a negligible function.

At step 2, A_i, B_i, R_i, S_i are transmitted. Here, M_i, N_i, r_i, s_i are independent random matrices with each element uniformly distributed over a specified range. Therefore, A_i and B_i also have a uniform distribution and hence, A_i, B_i are computationally indistinguishable from M_i, r_i, N_i, s_i.

On the other hand, R_i follows the distribution of r_i according to the ECDLP assumption of the ECC encryption algorithm. Thus, R_i is also computationally indistinguishable from r_i. The same holds true for S_i and s_i. From this, we can see:

$$|\Pr[A(A_i, B_i, R_i, S_i, P, Q, C, D) = 1] - \Pr[A(M_i, N_i, r_i, s_i, p_i, q_i, c_i, d_i) = 1]| < \mu(n).$$

with $\mu(n)$ being a negligible function.

At the final step, each client sends $T_i = W_i + M_i N - M N_i$. Since M_i and N_i are matrices where each element is independent and uniformly distributed, T_i also has a uniform distribution. Therefore, T_i, M_i, and N_i are computationally indistinguishable.

Hence, the proposed protocol guarantees the privacy of each participant's input in the actual model.

Subsequently, the thesis establishes that the multi-party secure sum protocol preserves the privacy of honest users, even in the case where $n-2$ out of n users, who are assumed to be semi-honest, collude.

To demonstrate that the protocol ensures the privacy of honest parties against collusion involving up to $n-2$ semi-honest parties and the computation party, one must introduce a simulator M. This simulator should be able to replicate what the dishonest parties and the computation party observe during the execution of the protocol using a polynomial-time algorithm. Specifically, it is crucial to provide a polynomial-time algorithm that generates the joint view of the computation party and the dishonest parties based solely on the knowledge of the semi-honest parties, the outputs, the public keys, and a set of ECC ciphertexts on the elliptic curve field $E(\mathbb{Z}_\mathbf{q})$.

Without loss of generality, we assume that two clients U_1 and U_2 do not collude, while the central server and the other clients U_i, for $i \in I = \{3, 4, \ldots, n\}$, collude with each other. In the secure protocol, each client sends only the encrypted value T_i, public shared values A_i, B_i, R_i, S_i, and public keys P_i, Q_i, C_i, D_i to the server. The public keys P_i, Q_i, C_i, D_i are uniformly random because the corresponding private keys

p_i, q_i, c_i, d_i are also uniformly random. Similarly, A_i, B_i, R_i, S_i are uniformly random values because $M_i, N_i, r_i, s_i, q_i, p_i, c_i, d_i, P, Q, C, D$ are uniformly random.

To prove the theorem, we must construct a probabilistic polynomial-time algorithm that can simulate the computation of the messages W_1 and W_2 using only the final sum T, the corrupted clients' knowledge $\{p_i, q_i, c_i, d_i, M_i, N_i, r_i, s_i, W_i\}$, and the public values $P_1, Q_1, C_1, D_1, A_1, B_1, R_1, S_1$, along with the shared public values: P, Q, C, D, R, S, M, N.

We denote the algorithm that satisfies the above assumption as M. Algorithm M uses (u_{11}, v_{11}), (u_{12}, v_{12}), (u_{21}, v_{21}), and (u_{22}, v_{22}) as follows:

$$(u_{11}, v_{11}) = (r_1 G + p_2 q_1 G, p_2 G) \tag{2.3.4}$$

$$(u_{12}, v_{12}) = (s_1 G + p_2 c_1 G, p_2 d_1 G) \tag{2.3.5}$$

$$(u_{21}, v_{21}) = (r_2 G + p_1 c_2 G, p_1 d_2 G) \tag{2.3.6}$$

$$(u_{22}, v_{22}) = (s_2 G + p_1 q_2 G, p_1 G) \tag{2.3.7}$$

To start, we compute the values of (R_1', S_1') and (R_2', S_2') as follows:

$$R_1' = u_{11} + \sum_{i \in I} p_i Q_1 + \left(r - \sum_{i \in I} r_i \right) G - \left(u_{21} + \sum_{i \in I} q_i P_1 \right) \tag{2.3.8}$$

$$R_2' = u_{21} + \sum_{i \in I} p_i Q_2 + \left(r - \sum_{i \in I} r_i \right) G - \left(u_{11} + \sum_{i \in I} q_i P_2 \right) \tag{2.3.9}$$

$$S_1' = u_{12} + \sum_{i \in I} p_i C_1 + \left(s - \sum_{i \in I} s_i \right) G - \left(u_{22} + \sum_{i \in I} c_i P_1 \right) \tag{2.3.10}$$

$$S_2' = u_{22} + \sum_{i \in I} p_i C_2 + \left(s - \sum_{i \in I} s_i \right) G - \left(u_{12} + \sum_{i \in I} c_i P_2 \right) \tag{2.3.11}$$

Therefore:

$$\{\mathbb{M}(I, \bar{v}_I, f_I(\bar{v}), f(\bar{v}))\} = \left\{ [P_i, Q_i, C_i, D_i]_{i=1}^n, P, Q, C, D, [R_i, S_i]_{i=3}^n, [R_i', S_i']_{i=1}^2, s, r \right\}$$

and

$$\left\{ VIEW_{A,I}^{\Pi}(\bar{v}), \mathrm{OUTPUT}^{\Pi}(\bar{v}) \right\} = \left\{ [P_i, Q_i, C_i, D_i]_{i=1}^n, P, Q, C, D, [R_i, S_i]_{i=1}^n, s, r \right\}$$

Therefore, we can see that: $\left\{VIEW^{\Pi}_{A,I}(\bar{v}), \mathrm{OUTPUT}^{\Pi}(\bar{v})\right\}$ and $\{\mathbb{M}(I, \bar{v}_I, f_I(\bar{v}), f(\bar{v}))\}$ differ only in their $[R'_i, S'_i]^2_{i=1}$ and $[R_i, S_i]^2_{i=1}$ values.

Given that $r_i, s_i, q_i, p_i, c_i, d_i$ are randomly selected from $\mathbb{Z}^*_{\mathbf{q}}$, under the ECDLP assumption, the elements $u_{12}, u_{21}, v_{12}, v_{21}$ along with $r_i, s_i, q_i, p_i, c_i, d_i$ are computationally indistinguishable.

According to the definition, the random variables R'_1, R'_2, S'_1, and S'_2 are dependent on u_{12}, u_{21}, v_{12}, and v_{21}. These variables, in turn, are functions of the independent random variables r_i, s_i, q_i, p_i, c_i, and d_i, which are uniformly distributed as they are randomly selected from the set $\{1, \dots, \mathbf{q}-1\}$. Consequently, R'_1, R'_2, S'_1, and S'_2 are also uniformly distributed. This uniform distribution implies that R'_1, R'_2, S'_1, and S'_2 are indistinguishable. Therefore, even with additional knowledge about these variables, corrupted parties and the aggregation server gain no further information beyond the publicly shared parameters and the data derived from collusion.

According to Definition 1.3.4, the protocol is *semantically secure* against the collusion of the adversarial party and up to $n-2$ semi-honest parties.

On the other hand, the parameters shared among the parties are based on the discrete logarithm problem over the elliptic curse prime field $E(\mathbb{Z}_{\mathbf{q}})$. This problem currently belongs to the NP-hard class; therefore, reconstructing the secret parameters from the publicly shared parameters is also an NP-hard problem. Given the current computational capabilities, when the secret parameters are randomly chosen from $(1, p-1)$ in accordance with safety conditions, the protocol also ensures *computational security* based on the difficulty of the discrete logarithm problem in the elliptic curse prime field $E(\mathbb{Z}_{\mathbf{q}})$. $\square$

2.3.4. Performance evaluation

2.3.4.1. Computational cost

In this section, the thesis mainly focuses on the time complexity. To facilitate the discussion, the following notations in table 2.4 will be used.

We analyze the time complexity the proposed protocol by each phase:

Table 2.4: Notations of the time complexity of operations in the mask matrix protocol

Notation	Time required to perform
T_M^{IP}	A multiplication operation between a integer and a point on the elliptic curve.
T_A^{PP}	An addition operation between two points on the elliptic curve.
T_M^{PP}	A multiplication operation between two points on the elliptic curve.
T_S	The Shanks' baby-step giant-step algorithm.
T_M^{II}	A multiplication operation between two integers.
T_A^{II}	An addition operation between two integers.
T_A^{FF}	An addition operation between two float numbers.

- System preparation:

  In the system preparation stage, each client computes a public key by multiplying a integer matrix with a point on the elliptic curve. This performance takes each client $d \times d \times T_M^{IP}$ for each key. Since each client possesses two public keys, the time taken by each client to prepare both public keys is $2 \times d \times d \times T_M^{IP}$.

- Initialization phase:

  During this phase, the server calculates the public parameters P, Q, C, and D by summing the corresponding public keys of all clients, where each public key corresponds to a matrix of points. As a result, the time taken by the server in this phase can be expressed as $4 \times (n-1) \times d \times d \times T_{AP}$, where n is the number of clients, d is the dimension of the matrix.

- First computation phase:

  Each client generates their message consisting of four components, A_i, B_i, R_i, and S_i. Since A_i and B_i are obtained by adding two integer matrices then client spends $2 \times d \times d \times T_A^{II}$ in the process. R_i is obtained by multiplying an integer matrix with a point to get r_iG, and q_iP and p_iQ are obtained by element-wise multiplying two matrices of points. Thus, each client spends $d \times d \times T_M^{IP} + 2 \times$

$d \times d \times T_M^{PP} + 3 \times T_A^{PP}$ in computing R_i. Therefore, the total time spent by each client to compute A_i, B_i, R_i, and S_i is

$$2 \times d \times d \times T_A^{II} + 2 \times d \times d \times T_M^{IP} + 4 \times d \times d \times T_M^{PP} + 6 \times T_A^{PP}.$$

For server side, server first consumes $2 \times (n-1) \times d \times d \times T_A^{PP}$ for computing R and S. Server then takes $2 \times T_S$ for finding r and s that satisfy $rG = R$ and $sG = S$. Finally, server consumes $2 \times n \times d \times d \times T_A^{II}$ since performing operations among integer matrices.

- Second computation phase:
  For the client side, each client computes T_i which contains element-wise multiplication and addition operations on integer matrices. This computation requires total $2 \times d \times T_A^{II} + 2 \times d \times T_M^{II}$.

  For the server side, we takes $(n-1) \times d \times d \times T_A^{II}$.

Figures 2.5 and 2.6 offer a detailed assessment of the computational costs associated with the proposed protocol, revealing the demands placed on both the client and server sides. Figure 2.5 illustrates the average computation time required per client, and it is evident that the computational burden on each client is higher compared to the first protocol. This increased cost is primarily due to the need for clients to compute multiple key matrices and intermediate values. These operations involve significant matrix-based calculations, which are inherently more resource-intensive than the simpler computations in the first protocol.

In contrast, Figure 2.6 shows the time required for the secure aggregation phase at the server. Here, we observe that the secure aggregation process itself is relatively efficient because it mainly involves basic matrix addition and multiplication. These operations are generally fast, especially when compared to more complex cryptographic processes. However, while the secure aggregation phase on the server side benefits from this simplicity, it is important to recognize that the server's computational responsibilities extend beyond aggregation. Specifically, the server also needs to assist in computing the shared values R and S, which introduces a substantial computational overhead.

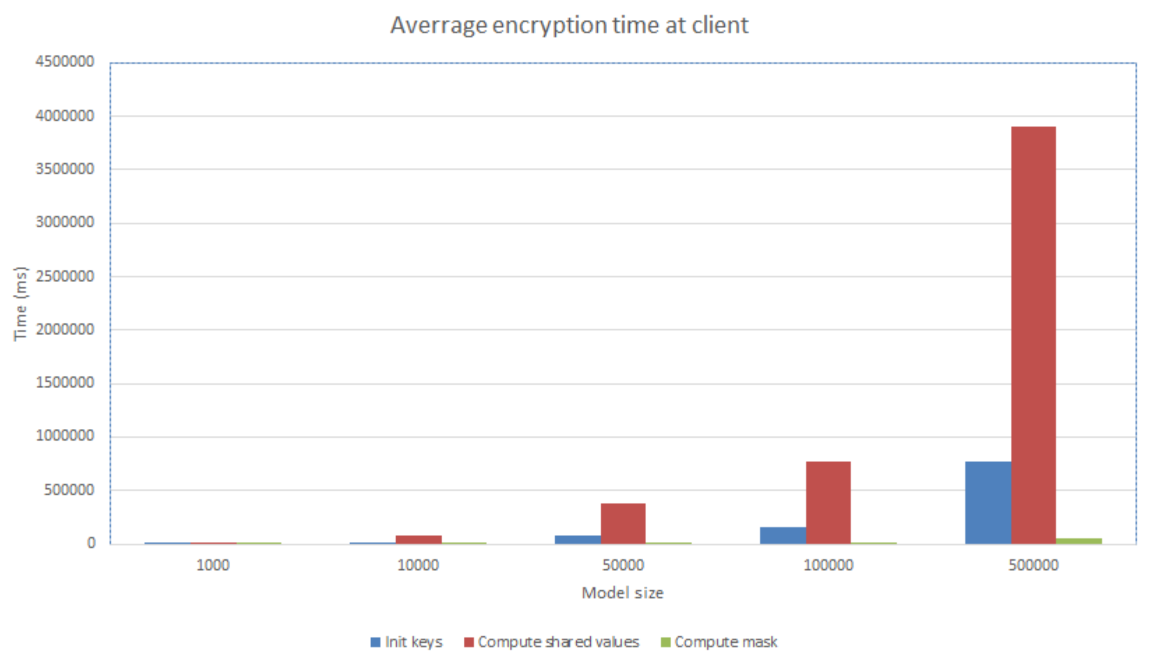


Figure 2.5: Average computation time at client with Secure multiparty sum vector protocol based on masked matrix

The calculation of these shared values requires the execution of the Shank algorithm, a computationally expensive process, particularly when large numbers or high levels of precision are involved. The Shank algorithm is a critical part of ensuring the security of the protocol, but its complexity adds significant overhead to the server's workload. As a result, the overall computational cost of the protocol is high on both the client and server sides.

On the client side, the increased complexity arises from the need to manage multiple key-related computations and the handling of intermediate results. On the server side, although the secure aggregation phase itself is relatively lightweight, the supporting operations for secure value sharing, particularly those involving the Shank algorithm, contribute to a significant increase in the overall computational burden.

In conclusion, while the protocol achieves its intended security and functionality, it imposes considerable computational costs. Clients face increased workloads due to matrix-based operations, and the server's role in secure value sharing adds to the overall complexity. These costs, while justifiable for the level of security achieved, indicate that the protocol requires careful consideration of computational resources,

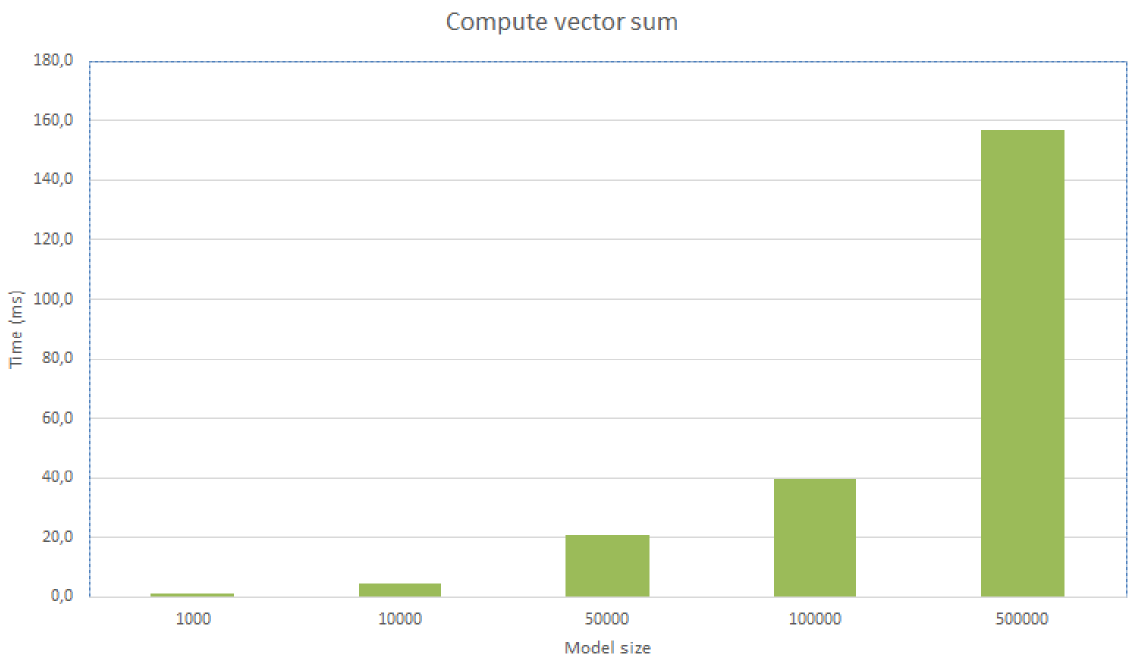


Figure 2.6: Secure aggregation cost at server with Secure multiparty sum vector protocol based on masked matrix

particularly in environments with limited processing power or bandwidth.

2.3.4.2. Communication cost

We denote the size in bits of a private message (W_i) as size$_m$ and the size of public keys in Elliptic Curve Cryptography (ECC) protocols as size$_k$. The messages exchanged between clients and the server can be classified into two groups according to their size:

- Messages with size size$_m$: A_i, B_i, M, N, T_i.
- Messages with size size$_k$: $P_i, Q_i, C_i, D_i, P, Q, C, D, R_i, S_i$

In the initialization phase, each client needs to send its public keys to the server. This process requires a total bandwidth of $4 \times \text{size}_m$ bits. The server then broadcasts P, Q, C, D to every client, which requires the server to establish n connections, each with a bandwidth of $4 \times \text{size}_k$ bits. Therefore, the total bandwidth for the initialization phase is $8 \times n \times \text{size}_k$ bits.

Moving on to the first computation phase, each client is required to send four messages (A_i, B_i, R_i, S_i) to the server. The transmission of A_i, B_i messages requires size_m bits of bandwidth for each message, while sending R_i, S_i requires size_k bits of bandwidth for each message. Additionally, the server sends M, N to each client, which incurs n connections with $2 \times \text{size}_m$ bits of bandwidth for each connection. Thus, the total bandwidth for this phase is $2 \times n \times \text{size}_m + 2 \times n \times \text{size}_k$ bits.

In the second computation phase, each client needs to send T_i to the server, which requires a bandwidth of size_m bits for each message. The total bandwidth for this phase is $n \times \text{size}_m$ bits.

For instance, with a private vector of 50,000 dimensions and an ECC encryption key of 256 bits (with sec256k1 curve), the total communication bandwidth during each round of the protocol is summarized in Table 2.5.

Table 2.5: Communication bandwidth of the proposed masking-based protocol with model size is 50,000 with 64bit precision and ECC key size if 256 bit (sec256k1)

	Client *i*	**Server**
Round 1	12.49MB (*not archive*) 6.2 MB (*archive*)	12.49MB $\times n$ 6.2 MB $\times n$
Round 2	7.7 MB 4.6 MB	**0.76 MB** $\times n$
Round 3	3.2 MB	0.38 MB $\times n$

Although the communication costs have increased significantly in comparison to the original federated learning framework, this increase is justifiable due to the stronger privacy and security guarantees offered by the new protocol. Despite the rise in communication overhead, the total costs remain substantially lower than those typically associated with secret-sharing-based protocols, such as those outlined in [162]. These protocols, while offering a high level of security, generally demand a far greater exchange of data between parties, which results in much higher communication overhead. In contrast, the proposed approach strikes a more effective balance between security and efficiency. It offers strong protection for sensitive information while keeping communication costs manageable, avoiding the excessive overhead often seen in other high-security protocols.

2.3.5. Discussion

The secure multiparty vector sum protocol based on the mask matrix offers a notable improvement in accuracy compared to the first protocol, which relies on integer quantization. The key strength of the second protocol is its ability to compute exact sums, avoiding the approximation errors that can arise in the first protocol. This level of accuracy is especially important in applications where even small deviations in the computed sum could lead to significant errors or inaccuracies in downstream tasks. However, this precision comes at the cost of increased computational overhead. Despite this, the second protocol still proves more efficient than the first when high precision is required. In the first protocol, as the need for accuracy increases, the Shank algorithm must be applied to much larger numbers, leading to an exponential increase in computational complexity. In contrast, the second protocol handles smaller noise values, which keeps computation time lower and more manageable. The computational complexity of the second protocol scales linearly with the size of the model and the number of participants, which is a significant improvement over the first protocol, where complexity grows exponentially with increased precision demands.

That said, the second protocol does have its drawbacks. While its linear complexity makes it more efficient in terms of scalability, it still incurs significant computational costs. Moreover, it lacks an integrated authentication mechanism, meaning that it does not ensure the security of communication between parties. This absence of built-in authentication requires the establishment of external authentication channels, adding complexity and cost to the system. These factors motivate the need for a more refined solution, which leads to the introduction of the third protocol in the dissertation. The third protocol not only retains the advantages of the mask matrix technique but also addresses its key weaknesses. By incorporating an authentication mechanism directly within the protocol, it eliminates the need for separate authentication channels, reducing both cost and complexity. This integrated solution is designed to provide a more balanced trade-off between security, efficiency, and computational overhead, and will be explored in detail in the next section of the dissertation.

2.4. Secure multi-party sum protocol using mask matrix with Authentication

2.4.1. Proposed protocol

The thesis proposes a protocol that utilizes the ElGamal scheme to accomplish a multi-party sum. This protocol leverages the advantage of homomorphic encryption, which enables multiple parties to collectively calculate the sum of their private messages while maintaining the confidentiality of their actual values. The security of the protocol is based on the difficulty of the Discrete Logarithm Problem. Additionally, the protocol demonstrates excellent performance when applied to floating-point numbers. The proposed protocol is summarized in Fig. 2.7.

Input:

- Each party U_i has private matrix $\overline{W}_i = [W_i^{(kj)}]; 1 \leq j,k \leq d$.
- Each party U_i has two private key matrices: $x_i = [x_i^{(kj)}], y_i = [y_i^{(kj)}]$.
- System parameters: Finite Field $\mathbb{Z}_\mathbf{p}$, generator g and invertible matrix H of size $d \times d$.

Output: Sum vector: $W = \sum_{i=1}^{n} W_i$.

Phase 1: Initialization Phase

- Each party U_i sends its public keys $\{X_i^{(jk)}\} = \{\mathbf{g}^{\mathbf{x}_i^{(jk)}}\}, \{Y_i^{(jk)}\} = \{\mathbf{g}^{\mathbf{y}_i^{(jk)}}\}$ to server.
- The server computes: $\{X^{(jk)}\} = \left\{\prod_{i=1}^{n} X_i^{(jk)}\right\}$; $\{Y^{(jk)}\} = \left\{\prod_{i=1}^{n} Y_i^{(jk)}\right\}$ and sends them back to all clients.

Phase 2: Main phase

- Each party U_i the public mask: $R_i^{(jk)} = g^{r_i^{(jk)}} \frac{X^{(jk)y_i^{(jk)}}}{Y^{(jk)x_i^{(jk)}}}$ and messages $T_i = v_i H + r_i$ then sends to the server.
- The server then computes $\{M_s^{(jk)}\} = \{\prod_{i=1}^{n} R_i^{(jk)}\}$ and find Q satisfy $g^{Q^{(jk)}} = M_s^{(jk)}$.
- The server obtains the vector sum by compute: $V = \sum_{i=1}^{n} v_i = (T - Q)H^{-1}$.

Figure 2.7: SVS Protocol based on mask matrix and ElGamal cryptosystem

In this section, we enhance the previously secure multi-party computation protocol by integrating an additional sub-protocol designed explicitly for user authentication. This sub-protocol leverages the robustness of the Schnorr signature based authentication mechanism, thereby strengthening the entire security protocol. This integration not only strengthens the trustworthiness of each participating user but also adds an extra layer of security to the entire computation process, ensuring a more secure and reliable system.

Initialization phase

Before beginning the protocol, several assumptions and requirements regarding the public parameters are shared between the server and clients and the client's public and private keys.

- *Preparation:* Each client holds a secret message containing a list of floating-point numbers. To conform to the protocol, we convert this message into a square matrix v_i of size $d \times d$ by arranging the numbers in the list into matrix rows. If the matrix is incomplete, we add 0s to fill it out.
- *Public Parameters:* We choose a cyclic group $G = Z_p$ where the Discrete Logarithm Problem is NP-hard, with p being a large prime number and g being a generator of G. We also select an uniform random invertible matrix H of size $d \times d$. The parameters H, p, and g are known to all participating clients. For simplicity in notation, operations within the Z_p field are assumed to be modulo operations.
- *Public keys, private keys and parameters of clients:* Each client U_i has two matrices of size $d \times d$ of secret keys, named $\{x_i\}$ and $\{y_i\}$. The elements of these matrices, $x_i^{(jk)}$ and $y_i^{(jk)}$, are selected randomly from $1, 2, \ldots, p-1$. Client U_i then computes their public keys by computing two matrices $\{X_i\}$ and $\{Y_i\}$ with each element is calculated by the formula $X_i^{(jk)} = g^{x_i^{(jk)}}$ and $Y_i^{(jk)} = g^{y_i^{(jk)}}$, where j is the row index and k is the column index of the matrix. In addition, client P_i creates a random mask matrix r_i with dimensions $d \times d$ with each element is selected randomly from $1, 2, \ldots, p-1$. This mask matrix will be removed during the aggregation process at the server.

- *Compute shared keys:* The participating clients U_i send their public key matrices $M_i = \{X_i, Y_i\}$ to the central server S. Then, the central server computes the X and Y matrices with entries $X^{(jk)}, Y^{(jk)}$ respectively:

$$X^{(jk)} = \prod_{i=1}^{N} X_i^{(jk)}; \quad Y^{(jk)} = \prod_{i=1}^{N} Y_i^{(jk)}$$

Then the server sends public matrices $M = \{X, Y\}$ back to each participating client.

Computation phase

The computation phase contains the following steps:

- Using their secret mask r_i, each participating client U_i computes the public mask R_i where, each element of R_i is computed by $R^{(jk)} = g^{r_i^{(jk)}} \frac{X^{(jk) y_i^{(jk)}}}{Y^{(jk) x_i^{(jk)}}}$, where each entries of the matrix $R_i^{(jk)}$ is computed based on the corresponding entries of the matrices $r_i^{(jk)}, X^{(jk)}, Y^{(jk)}, x_i^{(jk)}, y_i^{(jk)}$.

- Each client U_i computes a message $T_i = v_i H + r_i$.

- Then U_i, is tasked with the computation of a matrix signature, Sig_i. The following formula determines the entries of this matrix:

$$Sig_i^{(jk)} \equiv y_i^{(jk)} - x_i^{(jk)} \, H(Y_i^{(jk)} || T_i^{(jk)})$$

In this equation, $x_i^{(jk)}$ and $y_i^{(jk)}$ represent the elements of the private key matrices x_i and y_i respectively. $Y_i^{(jk)}$ is an element of the public key matrix Y_i, and $T_i^{(jk)}$ is an element of the encrypted message matrix T_i. As a result of this computation, client U_i sends three messages, namely R_i, T_i, and Sig_i, to the central server.

- At the central server, the server performs the calculation of the matrix M_s, where each entry is calculated by using the following formula

$$M_s^{(jk)} = \prod_{i=1}^{N} R_i^{(jk)}$$

From the matrix M_s, the server could calculate the sum of the masks Q such that each element $S^{(jk)}$ satisfies $g^{Q^{(jk)}} \mod p = M_s^{(jk)}$ by using the baby step-giant step algorithm.

- The central server then computes the sum of matrices T_i, represented as $T = \sum_{i=1}^{N} T_i$. Finally, the server obtains the sum of the private messages v_i of clients as

$$V = \sum_{i=1}^{N} v_i = (T - Q)H^{-1}.$$

- The central server initiates the user authentication process. This involves the computation of the matrix $Y_i^{'}$ and the verification of the equation $Y_i = Y_i^{'}$. The successful validation of this equation authenticates the client U_i. The detail formula of $Y_i^{'}$ is as follow:

$$\{Y_i^{'}\} = \{g^{Sig_i} X_i^{\gamma_i} \pmod{p}\}$$

  where $\gamma_i = H(Y_i||T_i)$.

2.4.2. Proof of correctness

In this section, we prove that the final output result V after performing the protocol is the sum of the private messages v_i of the participating clients.

Theorem 2.4.1. *The proposed protocol in the Figure 2.7 can calculate the sum of n vectors.*

Proof. We have:

$$T = \sum_{i=1}^{N} T_i = \sum_{i=1}^{N} (V_i H + r_i) = \sum_{i=1}^{N} (V_i H) + \sum_{i_i}^{N} r_i.$$

In order to simplify the notation, we will prove the theorem for each element individually while omitting the indices and only using symbolic representation for the matrix. It is important to remember that these are any arbitrary entries within the matrix. We prove $Q = \sum\limits_{i=1}^{N} r_i$. We have

$$M_s = \prod_{i=1}^{N} R_i = g^{\sum\limits_{i=1}^{N} r_i} \frac{X^{\sum\limits_{i=1}^{K} y_i}}{Y^{\sum\limits_{i=1}^{K} x_i}} = g^{\sum\limits_{i=1}^{N} r_i} \frac{\left(\prod\limits_{i=1}^{N} g^{x_i}\right)^{\sum_{i=1}^{N} y_i}}{\left(\prod\limits_{i=1}^{N} g^{y_i}\right)^{\sum_{i=1}^{N} x_i}}$$

$$= g^{\sum\limits_{i=1}^{N} r_i} \frac{\left(g^{\sum\limits_{i=1}^{N} x_i}\right)^{\sum_{i=1}^{N} y_i}}{\left(g^{\sum\limits_{i=1}^{N} y_i}\right)^{\sum_{i=1}^{N} x_i}} = g^{\sum\limits_{i=1}^{N} r_i} \frac{g^{\sum\limits_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}}{g^{\sum\limits_{i=1}^{N} y_i \sum_{i=1}^{N} x_i}} = g^{\sum\limits_{i=1}^{N} r_i}$$

Thus, if we obtain $Q^{(jk)}$ satisfying $g^{Q^{(jk)}} \mod p = M_s^{(jk)}$ then $Q^{(jk)} = \sum\limits_{i=1}^{N} r_i^{(jk)}$, or $Q = \sum\limits_{i=1}^{N} r_i$.

Thus, we have:

$$T = \sum_{i=1}^{N} (v_i H) + Q = \left(\sum_{i=1}^{N} v_i \right) H + Q$$

From the above equation, we can obtain

$$V = (T - Q) H^{-1} = \left(\sum_{i=1}^{N} v_i \right) H H^{-1} = \sum_{i=1}^{N} v_i .$$

$\square$

2.4.3. Privacy analysis

In this section, we analyze the security of the proposed protocol. We assume that all participating clients are semi-honest and comply fully with the protocol and the previously mentioned assumptions.

Theorem 2.4.2. *The multi-party sum protocol using El-Gamal is secure against any semi-honest participating clients.*

Proof. We consider three cases of data leakage from the protocol:

- Leakage of secret keys through key recovery attack.

- Leakage of mask matrix r_i.
- Leakage of the secret matrix v_i.

In the initialization phase, each client P_i sends their public key matrices X_i and Y_i with each element calculated from secret keys chosen at random with uniform distribution from $\{1,2,\ldots,p-1\}$. Due to the properties and assumption of the discrete logarithm problem, it is difficult to recover the secret keys from these corresponding public keys.

In the computation phase, each party only sends two messages, R_i and T_i. On the other hand, each entry of R_i has a formula.

$$R_i = g^{r_i}\frac{X^{y_i}}{Y^{x_i}}.$$

Determining the secret elements in this expression is equivalent to solving discrete logarithmic problems. As a result, the values of entries in the mask matrix remain secure and cannot be deduced from the public value sent out.

Considering the expression $T_i = v_iH + r_i$. Since r_i is a secret matrix, recovering the matrix v_i is equivalent to solving a system of linear equations $2 \times d \times d$ variables with only $d \times d$ equations. Since the elements of r_i are integers, the probability of predicting all these integer values with $\frac{1}{k^{d\times d}}$, where k is the selected integer space. Given a sufficiently large value of d and k, accurately reconstructing the secret matrix v_i from the public value is highly challenging, if not practically impossible. In other words, retrieving the value of the secret matrix v_i from the shared public value is impossible.

In short, from the shared public values, it is not possible for a passive attacker or any semi-honest client to recover the secret key matrices x_i, y_i, mask matrix r_i, and secret message matrix v_i of any participating client U_i. $\square$

Theorem 2.4.3. *The multi-party sum protocol using El-Gamal protects the confidential data of any participating client even if there are n-2 colluding members (and colluding with the server).*

Proof. To prove that the protocol protects the privacy of the honest parties against collusion by up to $n-2$ semi-honest parties and the computation party, one must present a simulator M that can simulate what the dishonest parties and the computation party can observe during the protocol execution using a polynomial-time algorithm. Specifically, it is necessary to exhibit a polynomial-time algorithm that computes the joint view of the computation party and the dishonest parties using only the knowledge of the semi-honest parties, the outputs, the public keys, and a number of ElGamal ciphertexts on the prime field $\mathbb{Z}_p$.

Without the loss of generality, we assume that two clients U_1 and U_2 do not collude while the central server and the others $U_i \| i \in I = \{3,4,\ldots,n\}$ collude with each other. In the secure protocol, each client only sends the encrypted value $V_i^{(j)}$ and two public keys $X_i^{(j)}, Y_i^{(j)}$ to the server. $X_i^{(j)}, Y_i^{(j)}$ are random values because the private keys $x_i^{(j)}, y_i^{(j)}$ are uniformly random. To prove the theorem, we must construct a probabilistic polynomial-time algorithm that can simulate the computation for the messages $\tilde{W}_1$ and $\tilde{W}_2$ using only the final sum $S^{(j)}$, corrupted clients' knowledge $\{x_i^{(j)}, y_i^{(j)}, V_i^{(j)}\}$ and public keys $X_1^{(j)}, Y_1^{(j)}, X_2^{(j)}, Y_2^{(j)}$.

We denote the algorithm that satisfies the above assumption as M. Algorithm M uses $(u_{12}, v_{12}) = (g^{\tilde{W}_1^{(j)}} g^{x_2^{(j)} y_1^{(j)}}, g^{x_2^{(j)}})$, $(u_{21}, v_{21}) = (g^{\tilde{W}_2^{(j)}} g^{x_1^{(j)} y_2^{(j)}}, g^{x_1^{(j)}})$ as its input to simulate $\tilde{W}_1^{(j)}, \tilde{W}_2^{(j)}$ as follows:

$$U_1^{(j)} = \frac{u_{12}.Y_1^{\sum_{i\in I} x_i^{(j)}} .g^{S^{(j)} - \sum_{i\in I} \tilde{W}_i^{(j)}}}{u_{21}.X_1^{\sum_{i\in I} y_i^{(j)}}}$$

$$U_2^{(j)} = \frac{u_{21}.Y_2^{\sum_{i\in I} x_i^{(j)}} .g^{S^{(j)} - \sum_{i\in I} \tilde{W}_i^{(j)}}}{u_{12}.X_2^{\sum_{i\in I} y_i^{(j)}}}$$

And so:

$$\left\{M(I, \bar{x}_I^{(j)}, f_I(\bar{x}^{(j)}))\right\} = \left\{ \left[X_i^{(j)}, Y_i^{(j)}\right]_{i=1}^n, X^{(j)}, Y^{(j)}, \left[V_i^{(j)}\right]_{i=1}^n, S^{(j)}, V^{(j)}, U_1^{(j)}, U_2^{(j)} \right\}. \tag{2.4.12}$$

Whereas:

$$\left\{VIEW_{A,I}^{\pi}(\bar{x}^{(j)})\right\} = \left\{\left[X_i^{(j)}, Y_i^{(j)}\right]_{i=1}^{n}, X^{(j)}, Y^{(j)}, \left[V_i^{(j)}\right]_{i=1}^{n}, S^{(j)}, V^{(j)}\right\}. \quad (2.4.13)$$

Therefore, we can see that: $\left\{VIEW_{A,I}^{\pi}(\bar{x}^{(j)})\right\}$ and $\left\{M(I, \bar{x}_I^{(j)}, f_I(\bar{x}^{(j)}))\right\}$ differ only in their $U_1^{(j)}, U_2^{(j)}$ values.

Given that $x_1^{(j)}, x_2^{(j)}, y_1^{(j)}, y_2^{(j)}$ are randomly selected from $\mathbb{Z}_p^*$, under the Decisional Diffie-Hellman (DDH) assumption, the elements $u_{12}, u_{21}, v_{12}, v_{21}$ along with $x_1^{(j)}, x_2^{(j)}, y_1^{(j)}, y_2^{(j)}$ are computationally indistinguishable.

According to the definition, U_1 and U_2 are dependent random variables determined by u_{12}, u_{21}, v_{12}, and v_{21}. These variables, in turn, depend on the independent random variables $x_1^{(j)}$, $x_2^{(j)}$, $y_1^{(j)}$, and $y_2^{(j)}$, which are uniformly distributed as they are randomly selected from the set $(1, p-1)$. Consequently, U_1 and U_2 are also uniformly distributed and thus thay and $x_1^{(j)}, x_2^{(j)}, y_1^{(j)}, y_2^{(j)}$ are indistinguishable. This implies that even with additional knowledge about U_1 and U_2, corrupted parties and the aggregation server gain no further information beyond the publicly shared parameters and the data derived from collusion.

According to Definition 1.3.4, the protocol is *semantically secure* against the collusion of the adversarial party and up to $n-2$ semi-honest parties.

We have: $T_i = v_i H + r_i$. Here, r_i is a matrix whose elements are randomly distributed uniformly within the range $[1, k]$. Therefore, T_i also has a uniform distribution and the probability of predicting the matrix T_i as well as r_i is uniformly distributed and equal to $\frac{1}{k^{d \times d}}$. Even if T_i, r_i, and v_i for $i = \{3, \ldots, n\}$ are known, the two remaining values of v_i cannot be distinguished with equivalent probability.

On the other hand, the parameters shared among the parties are based on the discrete logarithm problem over the prime field $\mathbb{Z}_p$. This problem currently belongs to the NP-hard class; therefore, reconstructing the secret parameters from the publicly shared parameters is also an NP-hard problem. Given the current computational capabilities, when the secret parameters are randomly chosen from $(1, p-1)$ in accordance with safety conditions, the protocol also ensures *computational security* based on the difficulty of the discrete logarithm problem in the finite prime field $\mathbb{Z}_p$. $\square$

2.4.4. Performance evaluation

2.4.4.1. Computational cost

In this section, the thesis mainly focuses on the time complexity. To facilitate the discussion, the following notations in table 2.6 will be used.

Table 2.6: Notation of operation time complexity in the Mask matrix with authentication protocol

Notation	Time required to perform
T_E	A exponential operation on $\mathbb{Z}_p$.
T_I	An inversion operation between two matrix.
T_M	A multiplication operation on $\mathbb{Z}_p$.
T_S	The Shanks' baby-step giant-step algorithm.
T_M^M	A multiplication operation between two matrix.
T_I^M	An inversion operation of matrix.
T_A^M	An addition operation between two matrix.

We analyze the time complexity the proposed protocol by each phase:

- Compute shared values phase:

  In the system preparation stage, each client computes a public key by performing exponential operation all entries of a integer matrix on prime field $\mathbb{Z}_p$. This performance takes each client $d \times d \times T_E$ for each key. Since each client possesses two public keys, the time taken by each client to prepare both public keys is $2 \times d \times d \times T_E$.

  The server calculates the public parameters X, and Y by multiplying the corresponding public keys of all clients, where each public key corresponds to a matrix of integer. As a result, the time taken by the server in this phase can be expressed as $2 \times (n-1) \times d \times d \times T_M$, where n is the number of clients, d is the dimension of the matrix.

Table 2.7: Time cost with different model size and number of clients

Number of clients	**Model Size**	**Time to Encrypt (s)**	**Time to Decrypt (s)**
5	100×100	2.4134	0.4651
	250×250	17.1407	2.8035
	500×500	60.6846	11.7244
10	100×100	2.5161	1.1419
	250×250	16.8426	5.5484
	500×500	60.7214	22.4258
20	100×100	2.5009	2.7467
	250×250	17.1056	11.2812
	500×500	60.6121	44.9112

- Secure aggregation phase:

  Each client generates their message R_i and T_i, where R_i costs $(3 \times T_E + T_I + 2 \times T_M)(d \times d)$ and T_i costs $T_M^M + T_A^M$

  For server side, server first consumes $(n-1) \times d \times d \times T_M$ for computing M_S. Server then takes T_S for finding Q that satisfy $g^Q = M_S$. Finally, server consumes $T_A^M + T_M^M + T_I^M$ since performing operations on matrices.

The Table 2.7 below illustrates the evaluation results of execution time based on the number of clients and the size of the model. It can be observed that the decryption time is extremely short and efficient due to the mask matrix being chosen from a range of small numbers.

Figure 2.8 presents the average encryption time per client during the public value sharing phase. Meanwhile, Figure 2.9 illustrates the time required for the secure aggregation phase. It is observed that the computation time on the client side is independent of the number of participants, assuming communication latency and server processing time are excluded. However, the aggregation time is influenced by the number of clients. As the number of clients increases, the time required to execute the Shank algorithm rises, consequently extending the overall computation time.

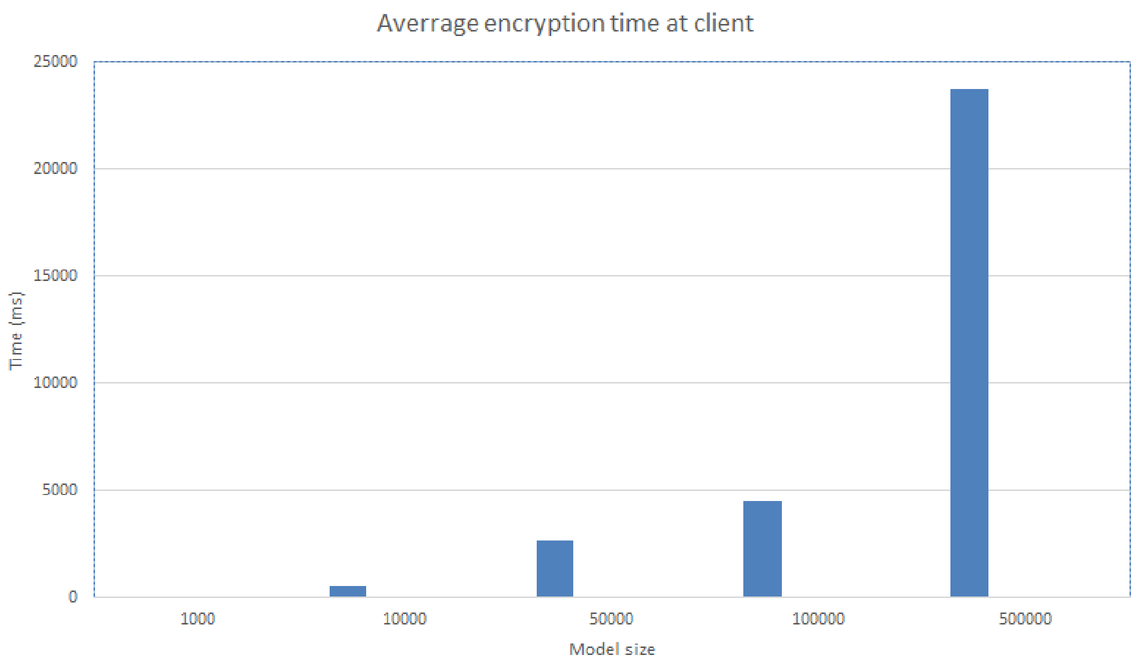


Figure 2.8: Avarage client encryption time with Secure multiparty sum with sub-authentication protocol

However, it should be noted that the execution time of this third protocol is significantly less than that of the first and second protocols. The reason is that, in the first protocol, to ensure calculation accuracy, the parameters must be chosen large enough, which leads to the Shank algorithm being performed on very large numbers. This becomes particularly challenging as the model size and the number of participants increase. In contrast, with the third protocol, we can select relatively small integers (typically not exceeding 100), greatly reducing the time required for the Shank algorithm's computations.

As for the second protocol, it involves the computation of four key values, and the Shank algorithm must be executed twice to determine the R and S values. This results in higher computational overhead compared to the third protocol.

2.4.4.2. Communication cost

We denote the size in bits of a private message (W_i) as size$_m$ and the size of public keys in Elgamal protocols as size$_k$.

In the initialization phase, each client needs to send its public keys to the server.

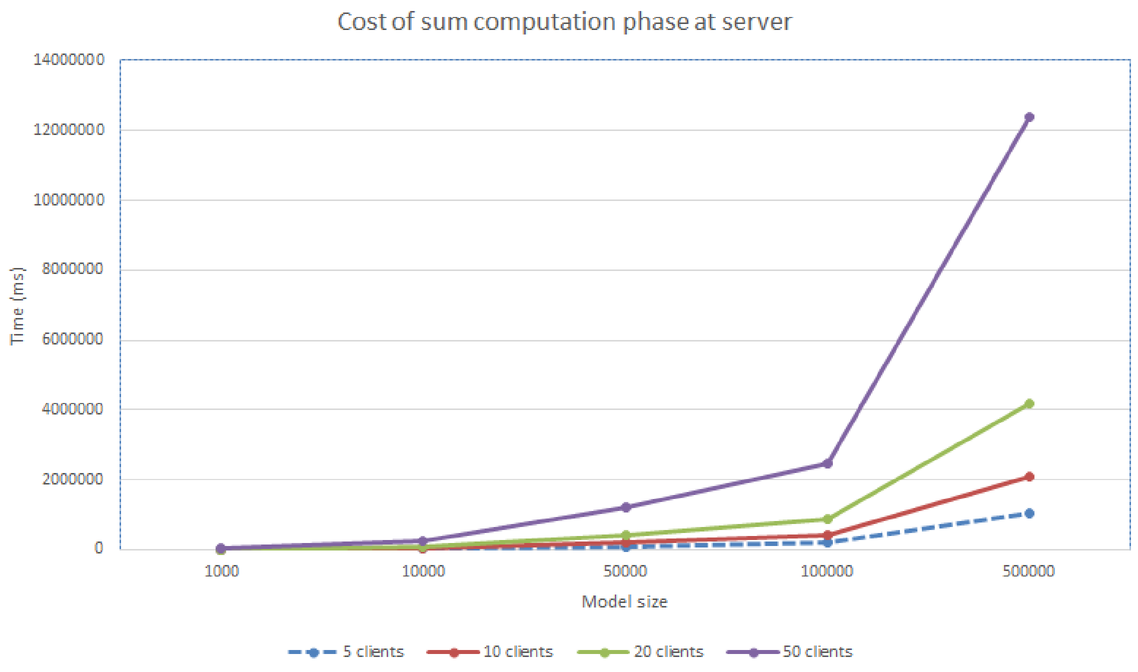


Figure 2.9: Time cost in aggregation phase with Secure multiparty sum with sub-authentication protocol

This process requires a total bandwidth of $2 \times d \times d \times \text{size}_k$ bits. The server then broadcasts X, Y to every client, which requires the server to establish n connections, each with a bandwidth of $2 \times d \times d \times \text{size}_k$ bits. Therefore, the total bandwidth for the initialization phase is $4 \times n \times d \times d \times \text{size}_k$ bits.

Moving on to the first computation phase, each client is required to send (R_i, T_i) to the server. The transmission of T_i messages requires size_m bits of bandwidth for each message, while sending R_i requires $d \times d \times \text{size}_k$ bits of bandwidth for each message. Thus, the total bandwidth for this phase is $n \times (d \times d \times \text{size}_k + size_m)$ bits.

For instance, with a private vector of 50,000 dimensions with 64bit precision and an Elgamal encryption key of 256 bits, the total communication bandwidth during each round of the protocol is summarized in Table 2.8.

Although it is clear that the overall communication cost has increased considerably compared to the original federated learning framework, this additional expense is warranted when taking into account the improved privacy and security assurances offered by the protocol. Notably, despite the rise in communication overhead, the total cost remains significantly more efficient than that of secret-sharing-based proto-

Table 2.8: Communication bandwidth of the proposed SMC with authentication protocol with model size is 50,000 and Elgamal key size is 256 bit

	Client i	**Server**
Round 1	3.07 MB	3.07 MB $\times n$
Round 2	*3.45 MB*	0.38 MB $\times n$

cols, such as those mentioned in [162]. While these protocols provide strong security, they often involve much higher data exchange between parties, resulting in notably higher communication expenses. In contrast, the approach proposed here strikes a better balance between security and efficiency, ensuring robust data protection without imposing excessive communication costs.

2.4.5. Discussion

The proposed secure multi-party summation protocol with authentication offers several key advantages that make it an attractive solution for large-scale privacy-preserving computations. This analysis will break down these advantages and trade-offs in greater detail, providing a deeper understanding of its potential compared to the other protocols.

First, computational time and bandwidth efficiency emerge as two crucial areas where this third protocol outperforms the second one, which relies on a masking matrix. In the context of secure multi-party computations, these improvements are vital because they directly influence the scalability of the system. By reducing the time it takes for clients to process their computations and decreasing the amount of data that needs to be transmitted between parties, the third protocol allows for faster and more efficient execution. This can be especially important in environments where real-time or near-real-time results are required, such as in distributed machine learning systems or federated learning scenarios.

However, when compared to the first protocol, this new approach introduces a higher communication cost. This means that more data must be exchanged between the parties during the execution of the protocol. Communication costs are a significant factor in secure multi-party computations because they often represent a bottleneck, especially in systems with limited bandwidth or high network latency. Nevertheless, the increase in communication costs in the third protocol is offset by improvements in other areas.

The first protocol, while lower in communication overhead, relies heavily on integer quantization techniques. Integer quantization is a method used to approximate real numbers by mapping them to a finite set of integers, which simplifies computation but can introduce errors. To maintain high accuracy, this protocol requires very large integers, which in turn necessitate expensive computational operations, such as handling large number arithmetic and performing operations over large finite fields. This increases the overall computational complexity and makes the protocol less efficient, particularly as the number of participants and the size of the model grow. Additionally, the use of large integers can introduce a loss of precision, which is especially problematic in sensitive applications where even small errors can propagate and lead to incorrect results.

In contrast, the third protocol achieves a more favorable trade-off between communication and computation. Although it incurs higher communication costs than the first protocol, it drastically reduces the computational overhead. This is particularly advantageous in systems where computation, rather than communication, is the primary limiting factor. For instance, in environments with powerful networks but limited computing resources on individual nodes (e.g., mobile devices in federated learning), the third protocol's computational efficiency would outweigh its increased communication cost.

A major strength of the third protocol is its built-in authentication mechanism, which allows participating parties to be authenticated within the protocol itself without the need for an external authentication channel. This is a critical improvement, as establishing external authentication channels can introduce additional complexity, overhead, and potential security vulnerabilities. By integrating authentication directly into the protocol, the third protocol simplifies the system's architecture and enhances its security. This built-in authentication also ensures that parties involved in the computation are verified securely, reducing the risk of malicious actors disrupting the process or tampering with the data.

Another significant potential enhancement of the third protocol is the ability to replace the Elgamal cryptosystem with Elliptic Curve Cryptography (ECC). Elgamal is a widely used cryptosystem that provides strong security guarantees, but it relies on large key sizes and expensive operations such as modular exponentiation. ECC, on the other hand, offers the same level of security with much smaller key sizes and more efficient cryptographic operations. By switching to ECC, the protocol could replace exponentiation operations with point addition on elliptic curves, which are computationally more efficient. This change would lead to faster execution times and reduced resource consumption, especially in environments with constrained computational power, such as IoT devices or mobile systems.

Additionally, ECC is known to be more secure against modern cryptographic attacks. The smaller key sizes used in ECC not only improve performance but also enhance security by making it harder for attackers to break the cryptosystem. This dual benefit of increased efficiency and enhanced security makes ECC an ideal choice for secure multi-party computation protocols that need to balance performance and robustness.

In summary, the third protocol presents a compelling option due to its balance of computation and communication costs, integrated authentication, and potential for optimization using ECC. While it has higher communication costs than the first protocol, these costs are mitigated by the significant reduction in computational overhead and the elimination of external authentication requirements. Furthermore, the possibility of switching from Elgamal to ECC presents a clear path toward further improving both the efficiency and security of the protocol. This makes the third protocol particularly well-suited for large-scale, distributed systems where privacy, efficiency, and security are of utmost importance.

2.5. Chapter Summary

This chapter offers an overview of the topic of SMC and comprehensively analyzes the most prominent prior research pertaining to this study. Based on the analysis results, three novel protocols have been proposed for privacy-preserving federated learning. These protocols include secure multi-party sum with decimal compression, secure multi-party floating point real number vector with masking matrix and a variant of ECC, and secure multi-party floating point real number vectors without needing pre-established secure/authenticated channels. Empirical evidence substantiates the assertion that the proposed protocols exhibit high security inside the semi-honest paradigm. The evaluations also demonstrate its effectiveness. Hence, the aforementioned protocols have the potential to be implemented in real-world scenarios.

CHAPTER 3. DEVELOPING SECURE FEDERATED LEARNING SCHEME BASED ON PROPOSED SECURE MULTI-PARTY SUM PROTOCOLS

In this chapter, the thesis delves into the integration of the protocols introduced in Chapter 2 within the framework of Federated Learning models. Specifically, it advocates for the application of secure multi-party computation (MPC) protocols tailored for real-number operations during the training phase of Federated Learning, applicable to both centralized and decentralized network environments. A comprehensive evaluation is conducted across various datasets and deep neural network architectures to demonstrate the robustness and efficiency of the proposed protocols.

3.1. Secure federated learning framework with semi-trusted aggregator server

3.1.1. Proposed framework

In this setting, a central server orchestrates the collaboration among multiple clients. The server's primary role is to aggregate the local models generated by individual clients, typically utilizing algorithms like Federated Averaging (FedAvg) to compute a unified global model. Once the global model is updated, the server redistributes it to all participating clients, ensuring a continuous and collaborative learning process. Thus, the server acts as a central hub, facilitating the exchange of model parameters among clients while safeguarding the privacy of each client's data.

The operational specifics of this setting are detailed in Framework 1.

Framework 1: Federated learning framework with semi-trusted aggregator server

Input: A semi-trusted aggregator server and set of n clients $\mathscr{U} = U_1, U_2, \ldots, U_n$ with private datasets D_i of sizes m_i.

F: Fraction of clients participating per communication round.

Hyperparameters: T (number of communication rounds), E (number of local epochs), B (local mini-batch size), W^0 (initial global model).

Output: Trained global model W.

Training Procedure:

The training phase entails T communication rounds. Each round, denoted as t, comprises the following operations:

- The server selects $n_t = F \times n$ clients for participation in the current training round.
- **Client-side operations (executed by n_t clients in parallel)**:
  - Each client U_i trains the model W^t on its data D_i over E epochs, yielding W_i^{t+1}.
  - Client U_i transmits W_i^{t+1} to the server.
- **Server operations**:
  - The server combines the received models according to the Fed-Avg algorithm:

$$W^{t+1} \leftarrow \sum_{i=1}^{n} \frac{m_i}{M} W_i^{t+1}.$$

  - The updated global model W^{t+1} is then sent to all clients.

In this model, the direct transmission of local model parameters from each participant to the central server introduces the risk of inversion attacks, which could potentially compromise the privacy of the participants' training data. To address this vulnerability, the thesis proposes the adoption of three secure multiparty vector sum protocols, as outlined in Chapter 2, to safeguard the sharing of local models. These protocols effectively replace the traditional parameter aggregation process described in Framework 1. By leveraging these protocols, participants can collaboratively compute a global model in a manner that ensures the privacy of their individual contributions. The enhanced training framework, incorporating these protective measures, is detailed below in Framework 2.

Framework 2: Secure federated learning framework with semi-trusted aggregator server

Input: A semi-trusted aggregator server and set of n clients $\mathscr{U} = U_1, U_2, \ldots, U_n$ each with a corresponding private dataset D_i of sizes m_i.

F: Fraction of clients participating per communication round.

Hyperparameters: T (number of communication rounds), E (number of local epochs), B (local mini-batch size), W^0 (initial global model).

Output: Trained global model W.

Training Procedure:

The training phase entails T communication rounds. Each round, denoted as t, comprises the following operations:

- The server selects $n_t = F \times n$ clients for participation in the current training round.
- **Phase (1) - Compute public share values:**
  - *Client-side (executed by n_t clients in parallel)*: Send the public values corresponding to the client's private values to the server.
  - *The semi-trusted aggregator server:* Compute the public shared values and distribute them, along with the global model W^t, to all clients participating in the round.
- **Phase (2) - Secure Sum Computation:**
  - *Client-side (executed by n_t clients in parallel):*
    - Trains the model W^t on its data D_i over E epochs, yielding W_i^{t+1}.
    - Transmits the masked W_i^{t+1}: $Mask(W_i^{t+1})$, after applying transformations based on secret values, to the server.
  - *The semi-trusted aggregator server*:
    - Execute the secure sum computation phase with $Mask(W_i^{t+1}), 1 \leq i \leq n_t$ to obtain the global model:

$$W^{t+1} \leftarrow \sum_{i=1}^{n} \frac{m_i}{M} W_i^{t+1}.$$

    - Send the updated global model W^{t+1} to all clients.

The operation of the framework is illustrated in Figure 3.1. Here, **Phase (1) - Compute Public Share Values** and **Phase (2) - Secure Sum Computation** correspond to the execution phases in the SMC protocols proposed in Chapter 2.

In this framework, the integration of the proposed SMC protocols is employed to obfuscate the information related to the parameters shared with the semi-trusted aggregator server. This approach effectively safeguards the privacy of the participants' local data without compromising the accuracy of the training process. The enhancement in security is significant, though it does come at the cost of increased computational and communication overhead. A comprehensive evaluation of the framework, including these trade-offs, will be discussed in the subsequent section.

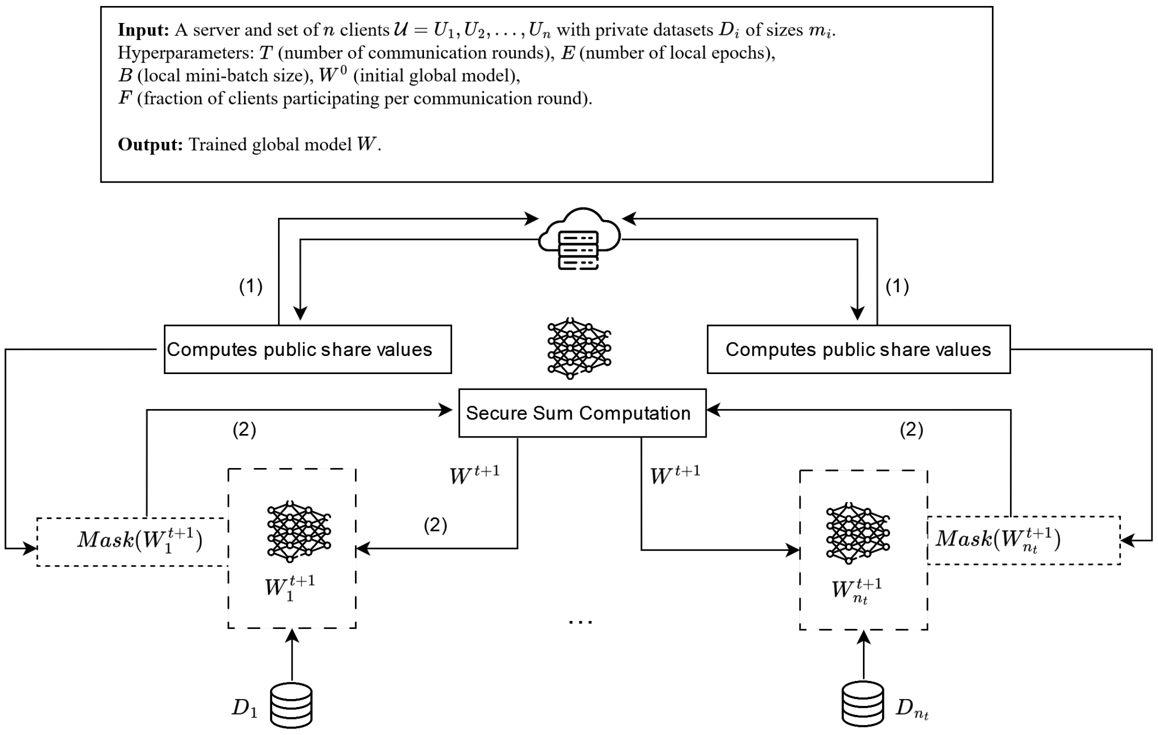

Input: A server and set of n clients $\mathcal{U} = U_1, U_2, \ldots, U_n$ with private datasets D_i of sizes m_i.
Hyperparameters: T (number of communication rounds), E (number of local epochs),
B (local mini-batch size), W^0 (initial global model),
F (fraction of clients participating per communication round).
Output: Trained global model W.
(1)
(1)
Computes public share values
Computes public share values
Secure Sum Computation
(2)
(2)
W^{t+1}
W^{t+1}
(2)
$Mask(W_1^{t+1})$
W_1^{t+1}
$Mask(W_{n_t}^{t+1})$
$W_{n_t}^{t+1}$
...
D_1
D_{n_t}

Figure 3.1: Secure federated learning framework with semi-trusted aggregator server

3.1.2. Experimental setup

3.1.2.1. Datasets

Three datasets, namely CSIC 2010, MNIST, and UCI SMS Spam Collection, are utilized to evaluate the effectiveness of the proposed framework.

- **CSIC 2010 Dataset**: This dataset is a widely recognized benchmark in web security, consisting of HTTP traffic data collected from a website. It includes both normal traffic and traffic generated by various web attacks, such as SQL injection, cross-site scripting (XSS), and command injection. The CSIC 2010 dataset contains over 36,000 HTTP requests, each labeled as either normal traffic or anomalous traffic. It is commonly used to evaluate the accuracy of machine learning models in detecting web attacks.
- **MNIST Dataset**: This dataset is a well-known benchmark for image recognition tasks, consisting of a large collection of 28x28 pixel images of handwritten digits, each labeled with the corresponding digit [189]. The MNIST dataset contains 60,000 training images and 10,000 test images, divided into ten classes

representing the digits from 0 to 9. It is frequently used to evaluate the accuracy and performance of machine learning models in image classification tasks.

- **UCI SMS Spam Collection**: This dataset consists of 5,572 English SMS messages, each labeled as either ”spam” or ”ham” (non-spam) [190]. It includes 4,826 ham messages (86.6%) and 746 spam messages (13.4%). The dataset is widely used to evaluate the performance of machine learning models in text classification tasks, particularly in distinguishing spam from non-spam messages.

3.1.2.2. Data Partitioning Strategies

In the experiments, two data partitioning strategies—IID (Independent and Identically Distributed) and non-IID—are used to distribute the dataset among multiple clients. The IID strategy involves randomly shuffling the entire dataset and dividing it into non-overlapping subsets, each with similar statistical properties, which are then distributed equally among the clients. This ensures that each client receives a balanced dataset that reflects the overall distribution of the entire dataset.

In contrast, the non-IID strategy employs a quantity-based label imbalance approach. Here, each client is randomly assigned a specific number of labels, and the corresponding data samples for these labels are distributed equally among the clients assigned those labels. This results in clients having data that is concentrated around specific labels, mimicking real-world scenarios where data distributions vary across different entities. This method ensures that no data samples overlap between clients, leading to diverse and specialized datasets for each client.

For instance, in the MNIST dataset, the IID partitioning would involve shuffling and equally dividing the digit images so that each client receives data containing all digit classes (0-9) with similar proportions. In the non-IID partitioning, however, clients might be assigned only a few digit classes, such as one client receiving data for digits 0 and 1, while another client has data for digits 7 and 8, leading to significant differences in label distribution across clients.

For the CSIC 2010 dataset, which has two labels (normal vs. anomalous requests), the IID partitioning would distribute a balanced mix of normal and anomalous HTTP requests to each client. In the non-IID scenario, some clients might predominantly receive normal requests, while others receive mostly anomalous requests, resulting in a specialized distribution of data across clients.

Similarly, with the SMS Spam dataset, which also has two labels (spam vs. ham), IID partitioning ensures that each client receives a balanced set of both spam and ham messages. However, under the non-IID strategy, some clients might receive mostly spam messages, while others might receive predominantly ham messages, leading to an imbalanced distribution across clients.

3.1.2.3. Neural Network Architectures

The federated learning framework are considering assumes that all participating clients agree on the same model architecture for training a global model. The thesis has built three distinct models to conduct experiments on the CSIC 2010, MNIST, and UCI SMS Spam Collection datasets.

- The Convolutional Neural Network (CNN) model employed for the MNIST dataset is meticulously architected around two fundamental blocks. Each block integrates a convolutional layer, succeeded by a ReLU activation function and a max-pooling operation, forming a robust feature extraction pipeline. The intricate features captured by these layers are then funneled into a fully connected layer, which is subsequently followed by a final fully connected layer with ten nodes to execute the classification task. The model's architecture, as detailed in Table 3.1, comprises a total of 155,606 parameters, reflecting the model's complexity and capability.
- The model chosen for the CSIC 2010 dataset is a Character-Level Convolutional Neural Network (CLCNN), meticulously designed to handle the intricacies of raw HTTP requests. After preprocessing, where URL strings are meticulously filtered from the HTTP requests, these strings are transformed into vectors of integers, which serve as the input to the model. The CLCNN architecture begins with an embedding layer that elegantly maps each integer to a 32-dimensional

Table 3.1: Model Summary of CNN architecture for MNIST dataset

Layer (Type)	**Output Shape**	**Number of Params #**
Conv2d (conv1)	[32, 1, 5, 5]	832
ReLU (relu1)	-	0
MaxPool2d (maxpool1)	-	0
Conv2d (conv2)	[64, 32, 5, 5]	51,264
ReLU (relu2)	-	0
MaxPool2d (maxpool2)	-	0
Linear (fc1)	[100, 1024]	102,500
ReLU (relu3)	-	0
Linear (fc2)	[10, 100]	1,010
Softmax	-	0
Total number of parameters		**155,606**
Trainable params		**155,606**
Non-trainable params		**0**

vector. This is followed by a series of two 1-D convolutional layers, each paired with max-pooling operations, to effectively capture the essential features of the data. The extracted features are then fed into a fully connected layer with 32 nodes, where a 50% dropout is strategically applied to enhance generalization. The final layer, a two-node fully connected layer equipped with a Softmax function, performs the classification, distinguishing between benign and anomalous requests. Table 3.2 provides a comprehensive summary of the model architecture, which comprises 50,242 parameters.

- The Long Short-Term Memory (LSTM) model designed for the SMS Spam Collection dataset is an advanced architecture that employs bidirectional LSTM layers to fully harness the contextual richness of text sequences. After the preprocessing stage, the input sequences—composed of integers representing word indices—are first transformed through an embedding layer, which maps these indices into dense vector representations. These embeddings are then fed into

Table 3.2: Model Summary of CLCNN for CSIC 2010 dataset

Layer (Type)	**Output Shape**	**Number of Params #**
Embedding	[70, 128]	8,960
Conv1d(conv1)	[64, 128, 3]	24,640
ReLU(relu1)	-	0
MaxPool1d(maxpool1)	-	0
Conv1d(conv2)	[64, 64, 3]	12,352
ReLU(relu2)	-	0
MaxPool1d(maxpool2)	-	0
Linear(fc1)	[64, 64]	4,160
ReLU(relu3)	-	0
Dropout	-	0
Linear(fc2)	[2, 64]	130
Softmax	-	0
Total number of parameters		**50,242**
Trainable params		**50,242**
Non-trainable params		**0**

two layers of bidirectional LSTMs, enabling the model to capture nuanced dependencies from both preceding and succeeding words in the sequence. The output from the final timestep of the second LSTM layer is funneled into a fully connected layer, where a softmax activation function generates a probability distribution over the possible output classes. This sophisticated model is meticulously crafted for text classification tasks, ensuring robust performance. The architecture's detailed summary is presented in Table 3.3, with a total of 1,009,282 parameters.

3.1.2.4. Computing Framework

The experiments were conducted on a desktop computer with the following specifications: an Intel Core i9-10900F CPU @ 2.80 GHz (10 cores, 20 threads),

Table 3.3: Model Summary of LSTM architecture for SMS Spam Collection dataset

Layer (Type)	**Output Shape**	**Number of Params #**
Embedding	[6972, 128]	892,416
LSTM(lstm_1)	-	66,560
tanh	-	0
LSTM(lstm_2)	-	50,176
Linear(fc)	[2, 64]	130
Softmax	-	0
Total number of parameters		**1,009,282**
Trainable params		**1,009,282**
Non-trainable params		**0**

32GB of RAM, and an NVIDIA GeForce RTX 3060 GPU with 12GB of GDDR6 VRAM, running on Windows 10 Pro 64-bit. The software environment for the implementation included Python version 3.11.9, along with several libraries: PyTorch version 2.3.0 for deep learning model development, gmpy2 version 2.1.2 for arbitrary-precision arithmetic, and ecdsa version 0.17.0 for elliptic curve cryptography. PyTorch is a flexible and widely-used machine learning library that provides dynamic computation graphs and GPU acceleration, enabling efficient training and experimentation of neural networks. The setup was chosen to ensure efficient processing and reliable performance throughout the training and evaluation of the proposed framework.

3.1.3. Experimental results and evaluation

3.1.3.1. The overall model performance

The thesis utilizes the datasets and the three corresponding network architectures discussed earlier to evaluate the impact of several factors on the performance of the global model. The specific objectives of the study are to:

- **Examine the impact of the number of clients involved in the training process**: This objective focuses on understanding how the varying number of clients participating in training affects the performance of the global model.

- **Evaluate the effect of dropout rates on client participation during each communication round**: This objective emphasizes the importance of assessing how different dropout rates influence the number of clients that remain active in the training process. In practical federated learning scenarios, client dropout may occur due to various factors such as network instability, limited computational resources, or lack of participation incentives.
- **Investigate the performance of the framework under varying levels of precision within a secure multi-party sum protocol using integer quantization techniques**: The goal is to assess the effectiveness of the protocol when client parameters are limited to different precision thresholds, thereby determining the trade-offs between communication efficiency and model accuracy.
- **Analyze the effect of data distribution across multiple clients on the global model's performance**: The study considers two scenarios for data distribution among clients: independent and identically distributed (IID) and non-IID. The objective is to evaluate how these different distributions impact the model's accuracy and generalization capabilities.

Note. In this section, the secure vector aggregation protocol using a masking matrix and the authenticated secure vector aggregation protocol both preserve the accuracy of the summation. Therefore, their accuracy will be evaluated together. However, the secure vector aggregation protocol that employs integer quantization introduces variations in accuracy. As a result, this protocol will be assessed separately, considering different choices of decimal precision. The detailed evaluation results are presented in the subsequent sections.

a. The impact of the number of clients involved in the training process

The first experiment examines the performance of the global model across different numbers of clients. Framework 2, incorporating two secure multiparty vector sum protocols based on masking matrices, was employed to train the model over 50 communication rounds. The figure below presents the outcomes of this study using three datasets: CSIC2010, MNIST, and SMS Spam Collection. The model's accuracy and loss metrics throughout the training process, tracked across each global epoch, are illustrated in Figure 3.2.

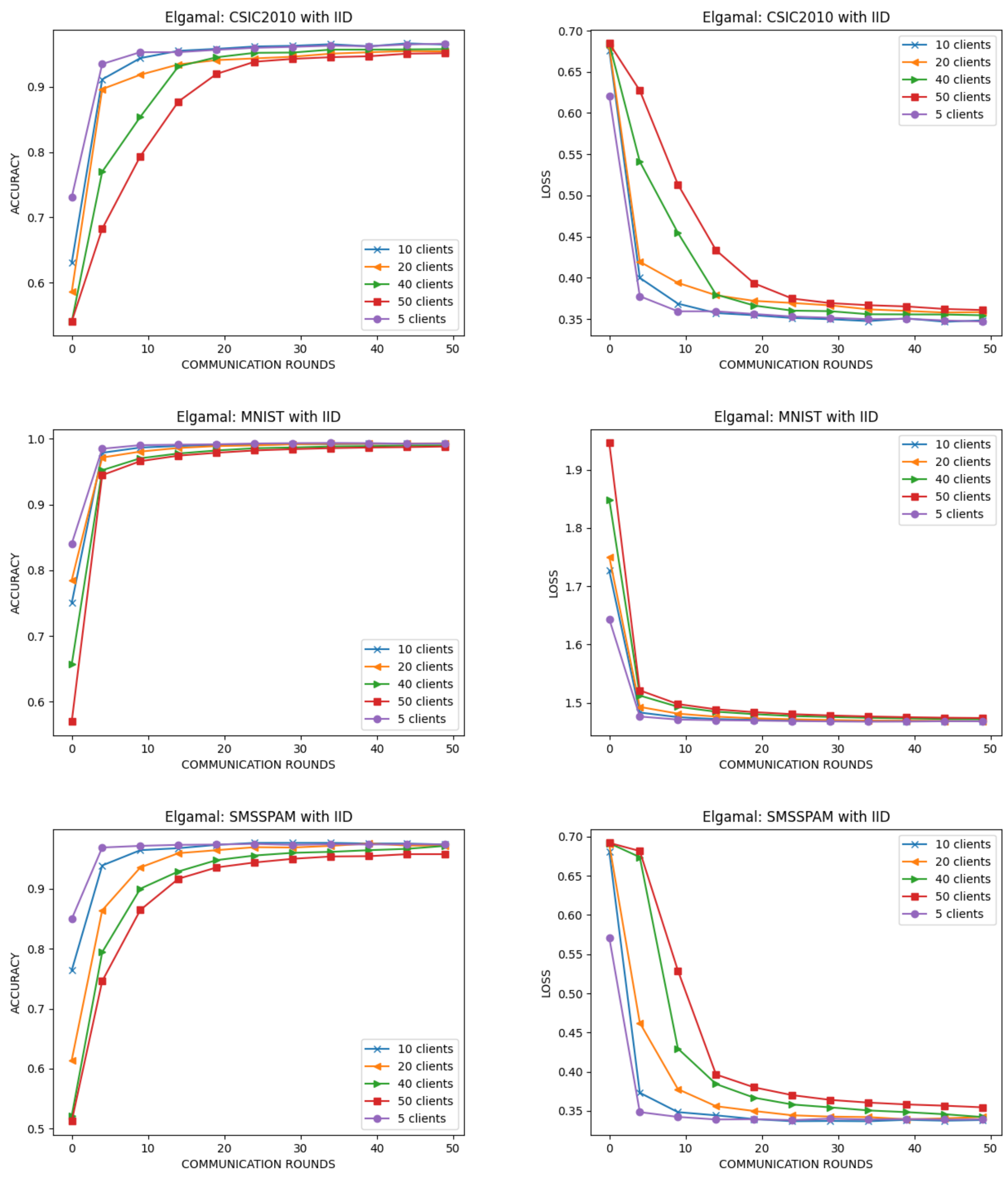


Figure 3.2: The results on accuracy and loss with different number of clients.

The experimental results from varying the number of clients (5, 10, 20, 40, and 50 clients) yield several insightful observations:

Firstly, as the number of clients increases, there is a noticeable decline in the initial accuracy of the global model. This outcome is anticipated, as a larger number of clients inherently reduces the amount of data each client contributes in the early stages, diluting the global model's initial learning process. For example, when only 5 clients are involved, the model achieves an initial accuracy of 0.7307. In contrast, with 50 clients, the initial accuracy drops significantly to 0.5403. This reduction can be attributed to the fact that with more clients, the global model's initial training is based on smaller, potentially noisier subsets of data, leading to a less accurate starting point.

Furthermore, the final performance of the model, measured after 50 communication rounds, generally improves when fewer clients are involved. This is likely because with fewer clients, each one contributes a more substantial share of data to the global model, thereby enabling more meaningful and robust local updates. These richer updates help the global model learn more effectively, leading to better overall performance. The disparity in final accuracy highlights the impact of data volume per client on the model's ability to generalize effectively.

However, an important dynamic emerges as the number of clients increases: the convergence rate of the global model slows down. This phenomenon is evident when examining the number of communication rounds required to reach a certain accuracy threshold. For instance, with only 5 clients, the model rapidly converges to an accuracy of around 0.95 within just 15 communication rounds. In contrast, with 50 clients, the model requires approximately 45 rounds to achieve a similar level of accuracy. The slower convergence can be explained by the increased heterogeneity and reduced data per client, which introduces greater variability in the model's updates, thereby necessitating more rounds for the model to stabilize and converge.

This trend is further complicated when working with unbalanced datasets, such as CSIC 2010 and SMS Spam Collection. Compared to a balanced dataset like MNIST, where labels are uniformly distributed, unbalanced datasets make convergence even more challenging. The uneven distribution of labels in such datasets exacerbates the difficulty of achieving consistent updates from clients, as some clients might contribute disproportionately to certain classes. This can lead to slower convergence and a more arduous path to reaching satisfactory accuracy levels. Despite these challenges, the model still achieves respectable final accuracies: 0.9923 for the MNIST dataset, 0.9654 for the CSIC 2010 dataset, and 0.9692 for the SMS Spam Collection dataset. This underscores the model's ability to ultimately converge to a high level of accuracy, provided sufficient training epochs and careful management of client contributions.

Moreover, the use of IID (Independent and Identically Distributed) datasets plays a critical role in ensuring a fair and meaningful comparison of model performance across different client configurations. By distributing the dataset in an IID manner, each client receives a representative sample of the entire dataset, which mitigates the risk of biased or skewed contributions to the global model. This balanced data distribution allows for reliable evaluation of the model's performance, irrespective of the number of clients involved, and ensures that the observed differences in accuracy and convergence rates are truly reflective of the underlying dynamics of the proposed framework rather than artifacts of data imbalance.

In conclusion, the experiment underscores the subtle yet significant impact of varying the number of clients within the proposed framework. Although an increase in the number of clients and the presence of unbalanced datasets may lead to slower convergence and reduced initial accuracy, the global model consistently converges to a satisfactory accuracy level given sufficient training rounds. The final accuracy levels—reaching 0.9923 for MNIST, 0.9654 for CSIC 2010, and 0.9692 for SMS Spam Collection—highlight the model's robustness. The proposed framework proves to be efficient, effectively balancing the trade-offs between client count, dataset balance, convergence speed, and overall model accuracy.

b. The effect of dropout rates on client participation during each communication round

A comprehensive series of experiments was undertaken to evaluate the performance of the proposed frameworks in a centralized setting with a semi-trusted aggregation server, utilizing two secure multiparty vector sum protocols based on masking matrices. The focus was particularly on assessing the impact of varying dropout rates. The objective was to understand how these dropout rates affect the accuracy and stability of the proposed frameworks. Systematic evaluations were conducted across a range of dropout rates—0%, 10%, 20%, 30%, 40%, and 50%. The resulting accuracy and loss metrics, tracked throughout the training process at each global epoch, are vividly illustrated in Figure 3.3.

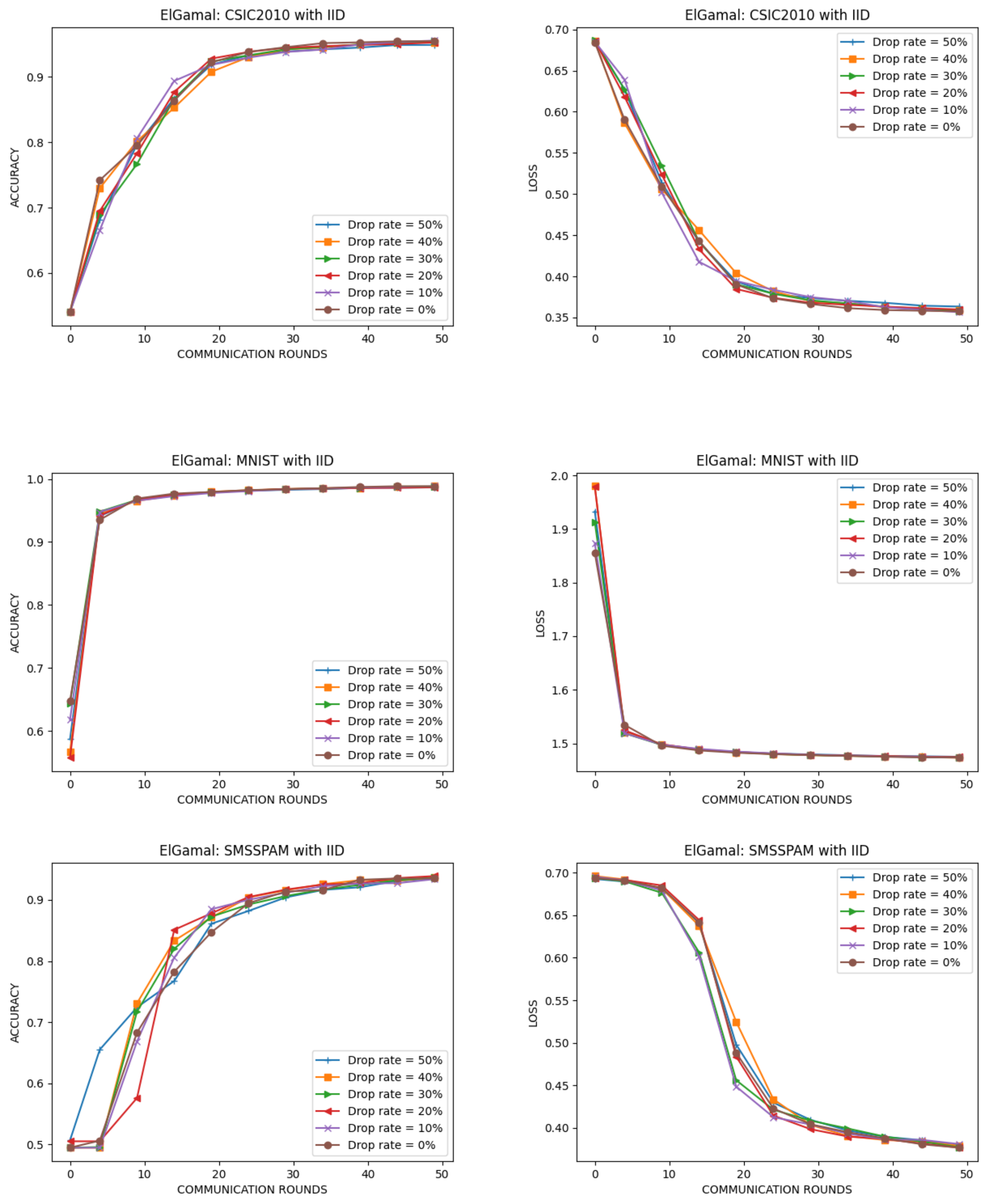


Figure 3.3: The results on accuracy and loss with different dropout rates.

Through extensive experimental analysis, the thesis has uncovered several critical insights into the effects of dropout rates on model performance. The results clearly indicate that the model's performance consistently improves as the number of training epochs increases, regardless of the dropout rate applied. This trend underscores the model's ability to effectively learn and adapt to the dataset over time, gradually refining its internal representations and improving its predictive accuracy. Interestingly, while it might be intuitively assumed that a 0% dropout rate—essentially training without dropout—would yield the best performance, the findings challenge this assumption. The analysis reveals that moderate dropout rates, such as 10% or 20%, can sometimes lead to superior outcomes. This highlights the role of dropout as a regularization technique that enhances the model's generalization capabilities by introducing a controlled level of randomness during training. By preventing the model from becoming too reliant on specific features of the training data, dropout reduces the risk of overfitting, thereby improving the model's ability to generalize to unseen data.

However, the analysis also cautions against the use of excessively high dropout rates. When dropout rates are too high, the model's convergence slows down, and its ability to generalize may be compromised. This is largely due to fewer clients participating in each round of global model updates, which results in less data being utilized in the training process. Consequently, the model may struggle to converge efficiently and may require more epochs to reach an optimal state. Additionally, with fewer clients contributing data, there is an increased risk of the model overfitting to the specific data subsets of the participating clients, further diminishing its generalization performance. This trade-off highlights the importance of carefully selecting dropout rates to balance convergence speed and model robustness. Notably, with unbalanced data, the model's convergence behavior is even more volatile and is more significantly impacted by dropout rates. The inherent imbalance in the data amplifies the sensitivity of the model to dropout, making it more challenging to find the optimal balance that ensures both convergence and generalization.

The interplay between IID (Independent and Identically Distributed) data and dropout rates provides additional nuance to this analysis. The results suggest that when data is IID, the negative effects of high dropout rates—such as slower convergence and reduced performance—can be somewhat mitigated. In an IID setting, each client's dataset remains representative of the overall population, ensuring that even with higher dropout, the global model updates still capture a broad and informative view of the data. This helps maintain performance levels by mitigating the risk of biased updates. However, it is crucial to recognize that even with IID data, extremely high dropout rates still pose challenges, as the model may experience slower convergence and reduced overall performance compared to scenarios with lower dropout rates. This underscores the need for careful consideration of dropout levels, even in favorable data conditions.

c. The performance of the framework under varying levels of precision within a secure multi-party sum protocol using integer quantization techniques

In this section, the thesis delves into the evaluation of the proposed framework utilizing the secure multiparty vector sum protocol with an integer quantization technique. This approach necessitates the encoding of the neural network's floating-point parameters into integer values, which are then processed by the SMC protocol. The central aim of this experiment is to rigorously investigate how varying levels of encoding precision influence the model's overall accuracy and performance. To achieve this, the framework was subjected to 50 communication rounds, with client parameters encoded to 3, 5, and 10 decimal places. These results were benchmarked against a control scenario where no quantization was applied, retaining the full precision of 15 decimal places.

As depicted in Figure 3.4, the results clearly show that reducing the precision of model parameters to just 3 decimal places leads to a substantial decline in accuracy. This is expected, as lower precision introduces significant quantization noise, which disrupts the model's ability to learn effectively from the data. On the other hand, higher precision settings—such as 5 and 10 decimal places—result in only minimal variations in performance compared to the control scenario. This suggests that moderate levels of quantization can be implemented without significantly compromising the model's accuracy.

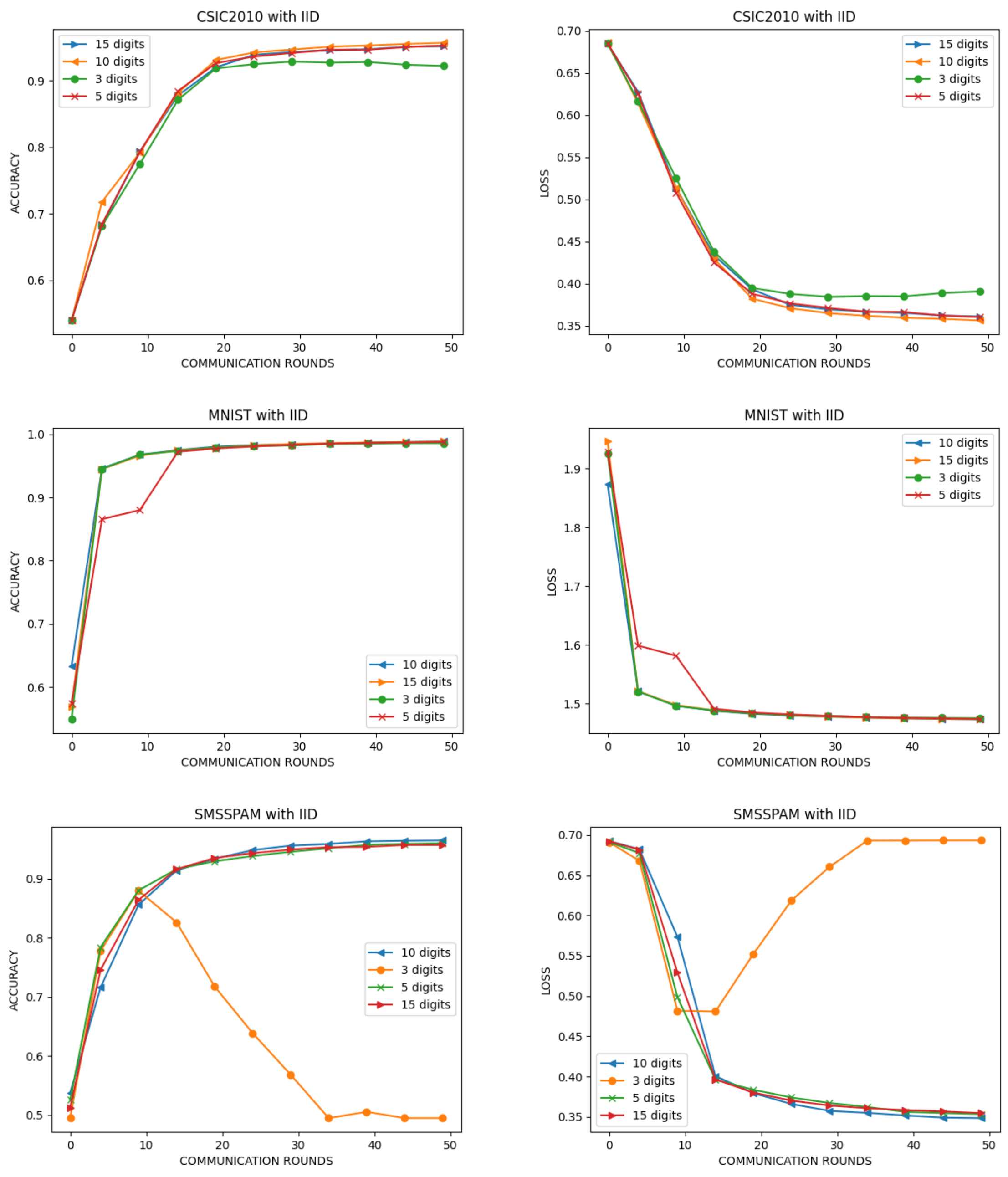


Figure 3.4: The result on accuracy for different precision levels.

An intriguing and somewhat counter-intuitive finding from this study is that using a precision of 10 decimal places actually resulted in slightly better accuracy than when no quantization was applied. This outcome can be attributed to the subtle introduction of randomness through quantization, which enhances the model's generalization ability. By incorporating this slight degree of randomness, the model is less likely to overfit to the training data, thereby improving its performance on unseen data. This phenomenon mirrors the effectiveness of dropout techniques commonly used in traditional neural networks.

However, the analysis also highlights the risks associated with excessively low precision. When only 3 decimal places are used, the noise introduced by quantization becomes too large, significantly degrading the model's accuracy. This effect is especially pronounced in the case of unbalanced datasets, such as the SMS Spam dataset, where the model's ability to generalize is further compromised. In such cases, the errors introduced by low precision exceed the model's capacity to effectively learn from the data, leading to a dramatic reduction in accuracy.

Conversely, using 5 decimal places strikes a balance where the quantization noise is small enough not to disrupt the model's learning process, allowing it to maintain accuracy close to that of the unquantized scenario. This finding underscores the importance of carefully selecting the level of precision in quantization. While a certain level of noise can be beneficial, enhancing the model's generalization ability, too much noise can be detrimental.

Moreover, it is important to consider the trade-offs associated with different levels of quantization. While lower precision can lead to faster computations and reduced storage requirements, it also increases the complexity of achieving convergence. The model may require more communication rounds to reach an optimal state, especially when dealing with unbalanced data. Additionally, the increased computational burden of managing higher precision, particularly in terms of time and processing power, grows exponentially. Thus, while the secure multiparty vector sum protocol with quantization can, in certain circumstances, offer superior accuracy compared to traditional federated learning methods without privacy guarantees, this comes at the cost of significantly higher computational expenses.

The analysis reveals that the use of quantization in secure multiparty computation introduces a delicate balance between enhancing model generalization and managing computational complexity. The choice of precision must be made carefully, taking into account the specific characteristics of the dataset, the number of participating clients, and the desired balance between accuracy and computational efficiency. These findings provide a nuanced understanding of the trade-offs involved and will be further explored in the subsequent sections of this thesis, where the computational costs associated with different levels of quantization are analyzed in greater detail.

Additionally, the accuracy of the model is heavily influenced by the number of participating clients. As highlighted in Chapter 2, the error in the summation process is directly affected by the number of clients involved. Recognizing this, the thesis also explores how varying the number of participants impacts the model's performance. The results of this analysis, specifically examining the scenario with 3 decimal places and different numbers of clients, are presented in Figure 3.5.

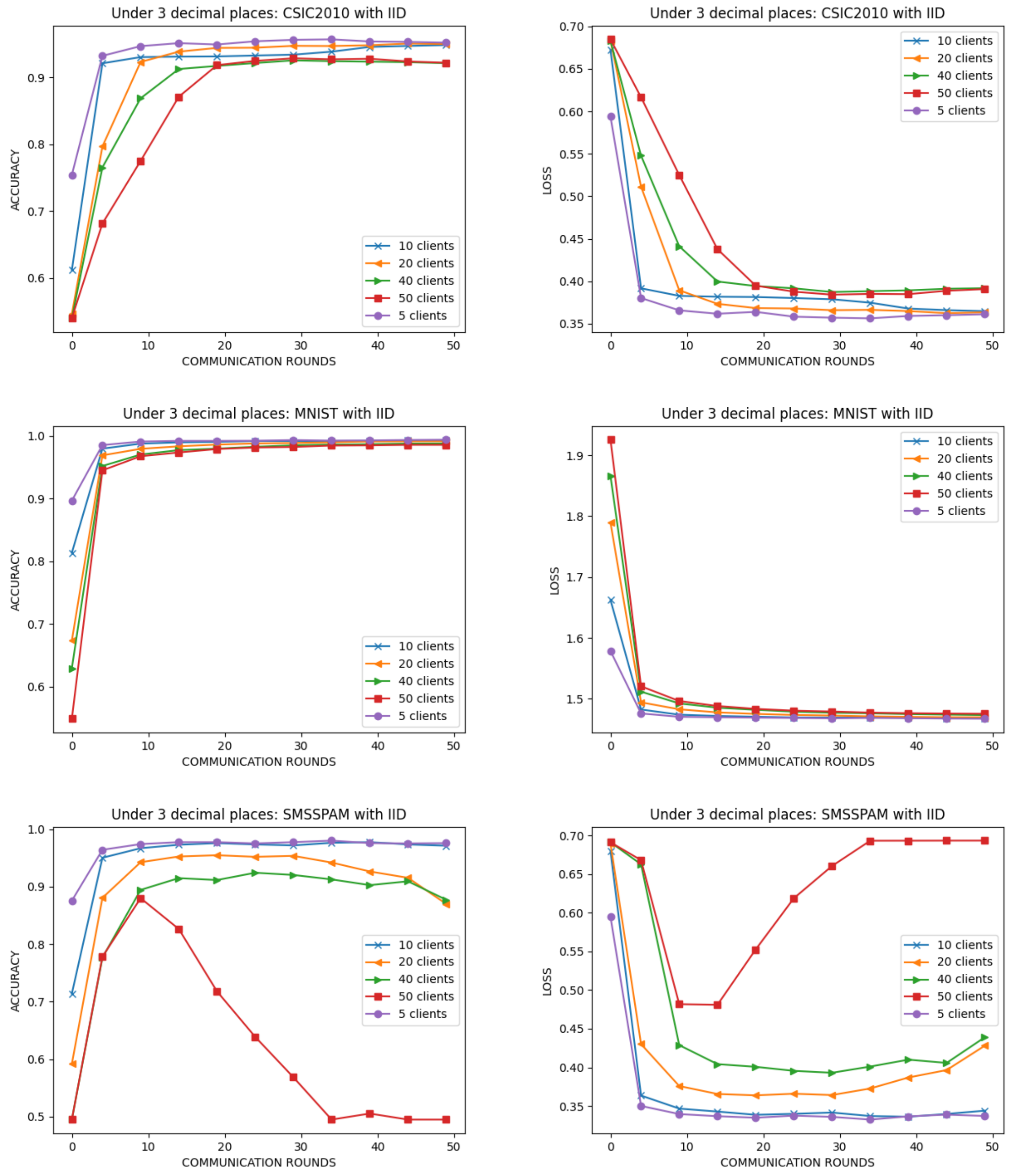


Figure 3.5: The accuracy with 3 decimal places when varying the number of clients.

The results reveal a clear and concerning pattern: as the number of clients increases, the model's accuracy steadily declines. This reduction in accuracy is primarily driven by the accumulation of computational errors during the parameter aggregation phase at the central server. As more clients contribute to the aggregation, the cumulative error grows, leading to a noticeable degradation in the model's performance. This issue is particularly severe when dealing with highly unbalanced datasets, such as the SMS Spam dataset. In such cases, when the number of clients reaches 50, the model's accuracy plummets dramatically, to the point where it nearly loses all predictive power. This sharp decline is likely due to the aggregation error surpassing a critical threshold, beyond which the model can no longer compensate, resulting in a significant loss of accuracy.

These findings underscore the critical balance required between the number of participating clients and the model's accuracy. While increasing the number of clients can theoretically improve the model's generalization by incorporating more diverse data, it also amplifies the aggregation errors, especially in scenarios involving unbalanced data. This trade-off is particularly pronounced in secure federated learning environments, where the precision of aggregated results is crucial. The analysis highlights the necessity of carefully managing the number of clients to maintain an optimal balance between model accuracy and the robustness of the aggregation process.

d. The effect of non-IID data distribution on the global model's performance

This dissertation aims to evaluate the accuracy of a training model within the context of Framework 2 when dealing with Non-IID (Non-Independent and Identically Distributed) data distribution. The primary focus is on understanding how the number of participating clients and their dropout rates impact the model's performance under different accuracy aggregation protocols. Specifically, the study examines two key protocols that utilize a masking matrix. Additionally, the dissertation investigates the effects of rounding precision within the protocol that involves vector quantization techniques for approximate summation on the overall accuracy of the model. The objective is to comprehensively assess how these factors—client number, dropout rates, and quantization precision—interact and influence the training outcomes in a distributed setting.

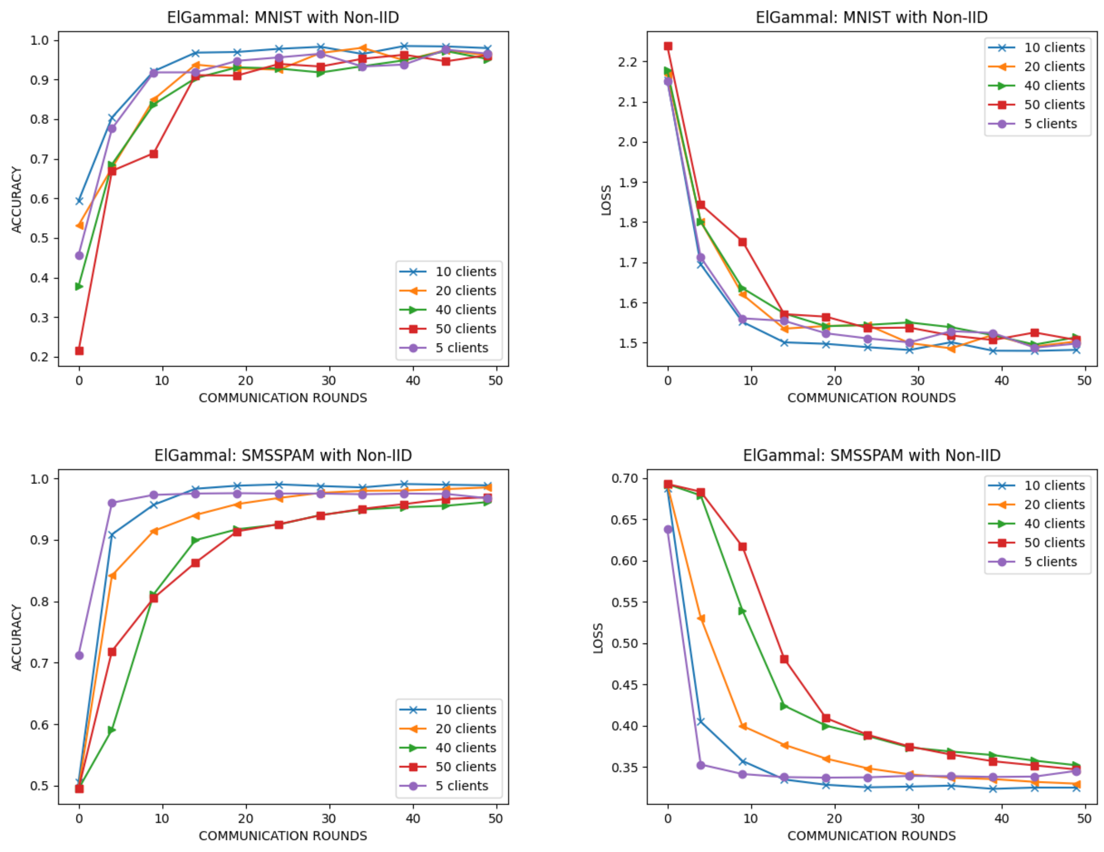


Figure 3.6: The results on accuracy and loss with different number of clients with Non-IID data distribution.

The findings, as illustrated in Figure 3.6, reveal that Non-IID data distribution can significantly increase the variance in model performance, particularly when the model is trained with a larger number of clients, such as 20, 40, and 50, as tested on the unbalance dataset (SMS Spam Collection). The results demonstrate that simply increasing the number of clients does not necessarily lead to better performance. Although a higher number of clients can enhance data diversity, this benefit is often outweighed by the increased complexity in coordinating clients and aggregating their updates. This complexity is particularly problematic in the scenario involving 50 clients, where the model exhibits difficulties in achieving consistent convergence. Conversely, a more moderate number of clients, such as the 10-client scenario highlighted in this thesis, strikes a better balance between data diversity and coordination complexity, leading to superior overall performance. Moreover, the study indicates that as the number of clients increases, the convergence rate of the global model tends to slow down, further emphasizing the trade-offs involved in managing larger distributed systems.

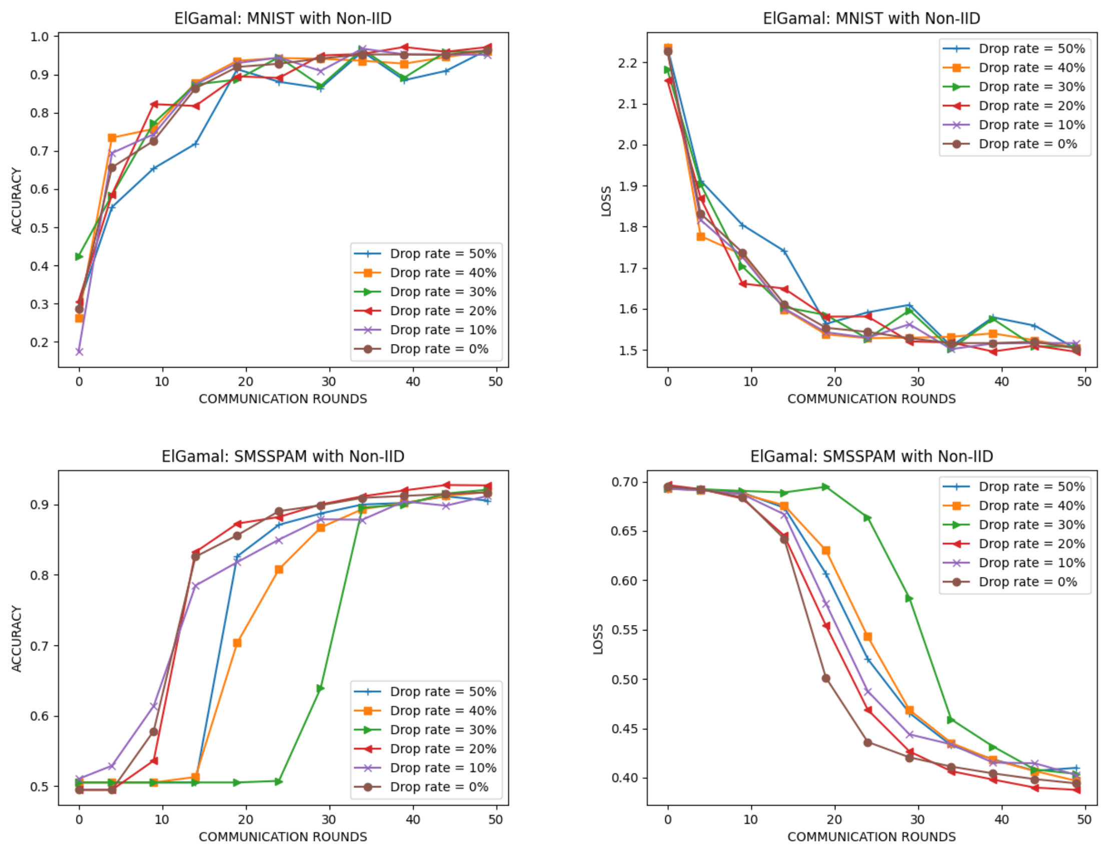


Figure 3.7: The results on accuracy and loss with different dropout rates with Non-IID data distribution

The evaluation of node dropout rates at each training round, as shown in Figure 3.7, provides significant insights into how dropout rates affect model accuracy, particularly in different types of datasets. When the dropout rate is high, there is a noticeable decline in accuracy, especially with unbalanced datasets like the SMS Spam Dataset. This decline can be attributed to the fact that in unbalanced datasets, certain classes are already underrepresented. A high dropout rate exacerbates this issue by further reducing the availability of data from these underrepresented classes during training. As a result, the model struggles to learn effectively, leading to poorer performance.

In contrast, when the dataset labels are balanced, the impact of dropout rates on model performance is much less pronounced. Balanced datasets ensure that all classes are equally represented, so even with some dropout, the remaining data still provide a sufficiently diverse and representative sample for the model to learn from. Therefore, the model's performance remains relatively stable despite variations in dropout rates.

The influence of dropout rates is particularly significant in the early rounds of training when the model is still forming its foundational understanding of the data. During these initial stages, if dropout is high, the model's learning process is disrupted, leading to a slower convergence and less accurate predictions. This is especially true for unbalanced datasets, where the early misrepresentation of classes can have a lasting negative impact on the model's ability to generalize well. Thus, it is crucial to carefully manage dropout rates and consider the balance of the dataset to ensure that the model can achieve optimal accuracy and robustness throughout the training process.

In this experiment, the neural network's floating-point parameters were encoded into integers to be compatible with the Secure Multi-Party Computation (SMC) protocol. This thesis thoroughly examines how the precision of this encoding influences the overall accuracy of the resulting model. Specifically, the study involved conducting experiments over 50 communication rounds, where client parameters were encoded to 3, 5, and 10 decimal places. The outcomes of these experiments were then compared to a control scenario where the parameters remained unencoded, retaining the full 15 decimal places of precision.

The results, as depicted in Figure 3.8, reveal a clear pattern: when the model parameters are rounded to just 3 decimal places, there is a notable drop in accuracy. This decline can be attributed to the significant loss of numerical precision, which hinders the model's ability to capture subtle variations in the data, leading to poorer performance. Conversely, when higher encoding precision is used—such as 5 or

10 decimal places—the impact on accuracy is minimal. This suggests that beyond a certain threshold, additional precision does not contribute significantly to model improvement, as the model can effectively learn and generalize even with slightly reduced precision.

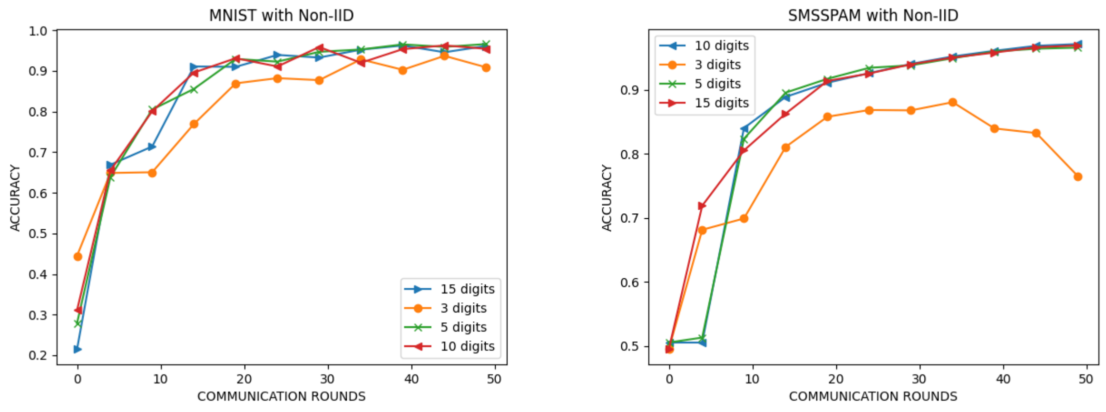


Figure 3.8: The result on accuracy for different precision levels with Non-IID data distribution

However, it's important to highlight that lower encoding precision not only diminishes accuracy but also slows down the convergence rate, requiring the model to take longer to reach optimal performance. This is because reduced precision can introduce more noise into the training process, making it harder for the model to stabilize during learning.

Interestingly, while these trends are consistent with those observed in IID (Independent and Identically Distributed) data distributions, there is a distinct difference in the non-IID scenario. In the IID case, 10 precision decimal consistently leads to better accuracy. However, in the non-IID context, the model's accuracy with 10 decimal places no longer surpasses that of the model without integer quantization. This could be due to the inherent challenges of non-IID data, where the uneven distribution of data across clients exacerbates the effects of even minor quantization errors. In non-IID scenarios, the diversity and imbalance in the data distribution may cause the model to be more sensitive to the loss of precision, making it difficult to achieve the same level of accuracy as in the IID case, even with higher decimal places.

This finding suggests that in non-IID settings, the benefits of higher encoding precision may be limited, as the interplay between data distribution and encoding precision becomes more complex. The model's performance might be more influenced by the distribution characteristics than by the encoding precision alone, highlighting the need for careful consideration of these factors when using the proposed framework.

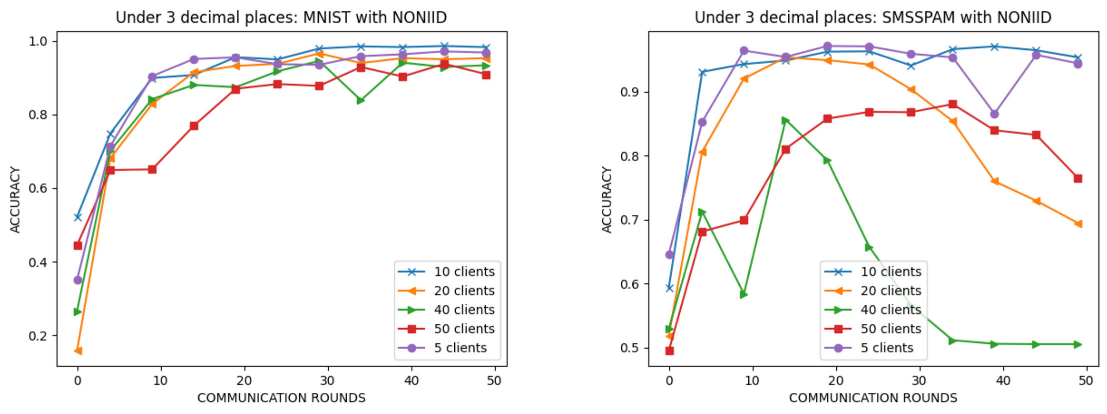


Figure 3.9: The accuracy value with 3 decimal places when varying the number of clients with Non-IID data distribution

When evaluating the impact of the number of participating clients on the protocol using integer quantization, the results reveal a more pronounced effect compared to scenarios with IID (Independent and Identically Distributed) data distribution. As shown in Figure 3.9, the model's accuracy significantly declines when only 3 decimal places are used, especially as the number of clients increases. While a similar trend is observed in IID cases, the effect is less severe. This pronounced accuracy drop becomes particularly evident when dealing with unbalanced datasets, such as the SMS Spam dataset.

The reason behind this can be attributed to the challenges introduced by both the limited precision and the increased number of clients in a non-IID setting. With more clients, especially in unbalanced datasets, the diversity in the data distribution becomes more extreme, leading to greater discrepancies in the local models trained by each client. When these local models are aggregated with reduced precision due to integer quantization, the errors introduced by the quantization process are amplified, particularly when the data distribution is uneven. This results in a more substantial degradation in overall model accuracy compared to the IID scenario, where the data distribution is more uniform, and the effects of quantization are less pronounced.

In summary, the impact of integer quantization is significantly more detrimental in non-IID settings, particularly with unbalanced data, where both the increased number of clients and reduced precision contribute to a marked decline in model accuracy. This highlights the need for careful consideration of quantization precision and client participation when using the proposed framework, especially when dealing with non-IID data.

3.1.3.2. Privacy

In the traditional landscape of distributed deep learning, we encounter a clear divide between data owners—those who possess valuable, often sensitive data—and the entities that drive the learning process, typically centralized servers or third parties. This separation presents significant privacy risks, as data owners must hand over their information to a learning party without having control over how it's used or what the final model looks like. They contribute their data but are disconnected from the actual learning and outcomes, leading to potential exposure of sensitive information during both the training and usage phases. This arrangement can lead to significant privacy dilemmas during both the development and deployment stages of the models.

In the training phase, when data owners submit their data to the learning server, there is always the risk that their raw information could be accessed directly by the server, compromised by attackers, or even handed over to authorities through legal or extralegal means.

In the usage phase, the problem persists. To use the trained model, data owners typically need to send new input data to the server that controls the model, risking exposure not only of their fresh data but also the results generated by the model.

These vulnerabilities underscore a fundamental problem: the lack of privacy and control over one's data in the existing distributed deep learning frameworks.

The framework developed in this thesis redefines privacy in the context of distributed deep learning, offering a novel approach to safeguarding sensitive information. It is meticulously designed to provide robust protection at every stage of the learning process, from initial training to final deployment. By ensuring the confidentiality of both input data and the resulting model outputs, this framework establishes a new standard for privacy preservation, addressing critical vulnerabilities and enhancing trust in distributed AI systems.

a. Addressing Direct Leakage

During the training phase, rather than requiring data owners to submit raw data, the proposed framework introduces an innovative mechanism where only encrypted intermediate model parameters are shared. This ensures that the actual training data remains securely stored on local devices, never exposed to the learning server or any external entity. By preserving full control over their data, owners can confidently contribute to the collaborative learning process without compromising their privacy, allowing them to participate in model training without the risk of sensitive information being revealed.

In the usage phase: Once the global model is trained, all participants gain access to it and can deploy it locally. This means they no longer need to send any data to a central server to get predictions. Both their input data and the model's outputs remain completely private, eliminating any risk of data leakage at this stage.

b. Addressing Indirect Leakage

Even in scenarios where attackers attempt to extract sensitive information by intercepting shared model parameters from participants, our framework employs advanced cryptographic protocols to neutralize such threats. These protocols facilitate the secure aggregation of model parameters, ensuring that no extra information is leaked, even in cases where up to $n-2$ parties collude. The security properties of this approach have been rigorously proven in Chapter 2, demonstrating its robustness in protecting privacy under adversarial conditions.

By adopting this approach, the proposed framework can effectively safeguard against both direct and indirect privacy threats, ensuring that data owners can participate in distributed learning without compromising the confidentiality of their information. The proposed framework represents a significant leap forward in privacy-preserving distributed deep learning. It empowers data owners by allowing them to keep their data private and under control throughout the training and usage phases, while still benefiting from the collective power of a global model.

3.1.3.3. Computational Cost

In the proposed protocol, each round involves the utilization of a semi-trusted aggregation server for parameter aggregation. Furthermore, during any training round, the necessity for joint participation is limited to n_t parties instead of all n parties, significantly reducing computational and communication costs within the protocol.

Given that time complexity is a crucial aspect of a training framework, it is denoted:

- T_i is the time for party i get the model updated at training round t.
- T_i^{train} is the time required for i-th party training his local model on his local data at training round t.
- T_i^{enc} is the time for computing shared parameters by Secure Vector Sum Protocol at training round t.
- T_i^{upload} is the time for uploading the masked models at training round t.
- $T_i^{download}$ is the time for downloading the global model from the master node at training round t.
- $T^{aggregation}$ is the time for aggregation server compute the final global model at each round.
- *nRound* is the total number of training rounds.

In a training round t, n_t parties parallelly train their local models, and the total time required for all of them to complete the protocol is given by

$$\max_{j=1}^{n_t}\{T_j^{train} + T_j^{enc} + T_j^{upload}\}. \tag{3.1.1}$$

The time complexity of the system for a party *i* is estimated using the following formula:

$$T_i = T_i^{train} + T_i^{enc} + T_i^{upload} + T_i^{download} + T^{aggregation} \tag{3.1.2}$$

Subsequently, the aggregation server computes and returns the global model to all parties. Thus, the time complexity for a party *i* to fully perform a training round *t* is computed using Equation 3.1.2.

Table 3.4: Comparison of the protocol protocols and datasets.

Protocol	**Dataset**	**Number of data samples**	**Number of Params**	**Avg. Local training time (s)**	**Avg. Compute share values time (s)**	**Avg. Aggregation time (s)**	**Total training time (s)**
SMC1 (3 digit)	CSIC2010	27610	50242	2.1	3.82	428.32	434.24
	MNIST	60000	155606	4.6	12.43	1568.34	1585.37
	SMS Spam	5572	1009282	15.5	89.16	11864.21	11968.87
SMC2	CSIC2010	27610	50242	2.1	746.27	0.29	748.66
	MNIST	60000	155606	4.6	2216.26	0.48	2221.34
	SMS Spam	5572	1009282	15.5	16031.31	3.72	16050.53
SMC3	CSIC2010	27610	50242	2.1	3.7	46.54	52.34
	MNIST	60000	155606	4.6	7.8	144.5	156.9
	SMS Spam	5572	1009282	15.5	71.5	942.62	1029.62

The table highlights the performance of three Secure Multi-party Computation (SMC) protocols—SMC1, SMC2, and SMC3—across different datasets: CSIC2010, MNIST, and SMS Spam. The local training time for all three protocols remains constant within each dataset. The most significant differences lie in the compute share values time and the aggregation time, which vary widely between protocols.

For example, SMC1 shows a reasonable compute share time for the smaller CSIC2010 dataset (3.82 seconds) but struggles significantly with the SMS Spam dataset, where it spikes to 89.16 seconds. Even more problematic for SMC1 is its excessive aggregation time on the SMS Spam dataset, ballooning to 11,864.21 seconds, making its total training time inefficient.

SMC2, while reducing aggregation time dramatically to almost negligible levels (0.29 to 3.72 seconds), introduces a significant compute share values time, especially on the SMS Spam dataset (16,031.31 seconds), which results in overall higher training times for larger datasets like SMS Spam.

On the other hand, SMC3 manages to optimize both compute share values and aggregation times. For the same SMS Spam dataset, SMC3 reduces the compute share values time to 71.5 seconds and aggregation to 942.62 seconds, resulting in a much lower total training time (1,029.62 seconds) compared to both SMC1 and SMC2.

In conclusion, while all protocols exhibit similar local training times, SMC3 stands out as the most balanced and efficient protocol overall, particularly for large datasets, due to its significantly reduced compute share values and aggregation times. This makes SMC3 the best choice for practical applications requiring efficient training across distributed parties.

3.1.3.4. Comparing the framework with other strategies

A comparative analysis was conducted to evaluate the accuracy of the proposed framework in relation to six alternative approaches. These include:

- **Standalone training:** In this scenario, each client trains their model utilizing only their local data, without any collaboration.
- **Centralized training:** This method involves the aggregation of clients' data at a centralized location, where the training process takes place.
- **Selective learning-based training:** Here, each client selectively shares a limited subset of their model's crucial parameters during the training process.
- **Generalized Average Federated Learning with differential privacy:** This approach is implemented in two variations, characterized by the inclusion of differential privacy and the application of either a small or large amount of noise.

The obtained results are shown in Table. 3.5.

The accuracy results across different datasets and data distributions highlight

Table 3.5: The accuracy comparison after 50 communication rounds.

Data distri-bution	**Dataset**	**Model**							
		Standalone	Centralized	Fed-Avg (Small noise)	Fed-Avg (Large noise)	Selective learning	Fed-Avg with El-Gamal	Fed-Avg with ECC	Fed-Avg with compress method
IID	MNIST	0.9658	0.9943	0.9741	0.9659	0.9629	0.9923	0.9923	0.9942
	CSIC 2010	0.9448	0.9676	0.9469	0.8745	0.9575	0.9654	0.9654	0.9685
	SMS Spam	0.9203	0.9806	0.9147	0.6397	0.9619	0.9692	0.9692	0.9728
Non-IID	MNIST	0.314	0.9943	0.9454	0.8954	0.9724	0.9869	0.9869	0.9882
	CSIC 2010	0.6103	0.9676	0.9442	0.7214	0.8534	0.9621	0.9621	0.9433
	SMS Spam	0.8821	0.9806	0.8173	0.5634	0.9662	0.9793	0.9793	0.9762

the effectiveness of various Federated Learning (Fed-Avg) approaches, particularly those involving cryptographic techniques and data compression. Under the IID data distribution, the Fed-Avg with compression method consistently performs at a level very close to centralized training, with accuracy rates of 99.42% for MNIST and 96.85% for CSIC 2010, indicating that efficient data communication strategies can nearly replicate the performance of centralized models. In comparison, Fed-Avg with ElGamal and Fed-Avg with ECC also show strong performance, with accuracy levels matching or closely following the compression method across most datasets. For instance, both Fed-Avg with ElGamal and ECC achieve 99.23% accuracy on MNIST and 96.54% on CSIC 2010 under IID conditions, demonstrating their capability to maintain high accuracy while ensuring data security through encryption. However, the compression method's slight edge suggests that reducing the data size without losing critical information may provide a marginal advantage over encryption techniques alone. This trend is consistent even in the Non-IID setting, where Fed-Avg with the compression method achieves an accuracy of 98.82% on MNIST, outperforming both ElGamal and ECC, which attain 98.69%. This indicates that while cryptographic methods like ElGamal and ECC are effective in securing data, the compression method may offer better efficiency in federated learning by minimizing communication overhead while maintaining high model performance.

Analyzing the above results leads to the following conclusions:

- **Impact of data distributions:**

  IID: In the case of IID (Independent and Identically Distributed) data, all three proposed methods outperform standalone training and provide competitive results to centralized training. This suggests that these methods can leverage the benefits of federated learning while maintaining good accuracy. The proposed methods also outperform Fed-Avg with small noise and large noise in most cases, highlighting their potential advantages in privacy-preserving federated learning.

Non-IID: The performance of the proposed methods generally decreases in non-IID scenarios. This is a common issue in federated learning due to the statistical heterogeneity of the data. Despite this, the proposed methods still outperform standalone training and Fed-Avg with noise, demonstrating their robustness to data heterogeneity.

- **Comparing to Standalone Training:**

  The results clearly indicate that all proposed methods outperform standalone training, especially in the non-IID scenario. This shows the advantage of collaborative training in federated learning, as it allows models to learn from a diverse range of data across multiple clients.

- **Comparing to Centralized Training:**

  Centralized training usually offers higher accuracy as it has access to all the data. However, it lacks privacy as all data needs to be aggregated in one place. Comparatively, the proposed methods provide competitive results, especially in the IID scenario. This indicates that three proposed methods can achieve near-centralized performance while preserving client privacy.

- **Comparing to Selective Learning-Based Training and Fed-Avg with Noise:**

  All three proposed methods generally outperform Selective learning and Fed-Avg with noise, demonstrating their effectiveness in privacy-preserving federated learning. They also provide strong privacy protection via encryption or data compression.

3.2. Secure federated learning framework in decentralized network settings

3.2.1. Proposed framework

Bottleneck in federated learning

Federated learning is a powerful approach for training global models on distributed data while preserving privacy. However, its reliance on a central server for the aggregation process has raised several concerns. A central server, the focal point for aggregating model updates, can inadvertently become a single point of failure and create performance bottlenecks due to the inherent limitations of centralized learning architectures. To address these issues, the thesis proposes a decentralized training framework that is designed to increase robustness and scalability.

Problem state

Consider a set of n distinct clients, with each client n possessing an exclusive local dataset denoted by D_i. These clients collectively partake in the training process of a universal deep learning or machine learning model pertinent to a specific problem domain. Notably, this process unfolds devoid of a central server functioning as an aggregator or as a focal point for communication.

Decentralized federated learning framework

In the proposed framework, there are n parties collaborating to train a global model. The process is structured as follows:

During each communication round, a master node is chosen randomly or via a voting scheme. At the local site, each client trains their local models on private data for E local epochs. Once the training is complete, clients send their models to the master node which is responsible for aggregating the received models. In an effort to minimize communication costs, the master node not only aggregates the models but also actively engages in the training process by contributing its local model to the global update. After the aggregation, the master node shares the updated global model with all clients, who proceed to the next communication round.

This decentralized approach significantly enhances the resilience and efficiency of federated learning systems. Distributing the aggregation responsibility and mitigating the risks associated with a single point of failure, this framework offers a promising alternative to traditional centralized learning architectures.

Framework 3: Decentralized federated learning framework

Input: Set of n clients $\mathscr{U} = U_1, U_2, \ldots, U_n$ with private datasets D_i of sizes m_i. F: Fraction of clients participating per communication round. Hyperparameters: T (number of communication rounds), E (number of local epochs), B (local mini-batch size), W^0 (initial global model).
Output: Trained global model W.
Training Procedure:
The training phase entails T communication rounds. Each round, denoted as t, comprises the following operations:

- Clients engage in a voting process to find a master node, U_{master}. The master node then selects $n_t = F \times n$ clients for participation in the current training round.
- **Client-side operations (executed by n_t clients in parallel)**:
  - Each client U_i trains the model W^t on its data D_i over E epochs, yielding W_i^{t+1}.
  - Client U_i transmits W_i^{t+1} to U_{master}.
- **Master Node Operations**:
  - U_{master} locally trains the model W^t on its data D_{master} over E epochs, producing W_{master}^{t+1}.
  - U_{master} combines the received models and its own model according to the Fed-Avg algorithm:
$$W^{t+1} \leftarrow \sum_{i=1}^{n_t} \frac{m_i}{m} W_i^{t+1} + \frac{m_{\text{master}}}{m} W_{\text{master}}^{t+1}$$
  - The updated global model W^{t+1} is sent to all clients.

Decentralized federated learning framework with secure sum protocol

To provide more details, these protocols are incorporated by integrating secure sub-modules into Framework 3, ensuring secure computations in each communication round. The secure multi-party summation protocol comprises two complementary sub-modules: an encryption sub-module used by the clients and a corresponding decryption sub-module employed by the master node. Consequently, a decentralized, federated learning framework supported by secure protocols is established. Here is a summary of the protocols:

Framework 4: Decentralized federated learning framework with secure multi-party sum protocol

Input: Set of n clients $\mathscr{U} = U_1, U_2, \ldots, U_n$ with private datasets D_i of sizes m_i. F: Fraction of clients participating per communication round. Hyperparameters: T (number of communication rounds), E (number of local epochs), B (local mini-batch size), W^0 (initial global model).
Output: Trained global model W.
Training Procedure:
The training phase entails T communication rounds. Each round, denoted as t, comprises the following operations:

- Clients engage in a voting process to find a master node, U_{master}. The master node chooses $n_t = F \times n$ clients for the current round.
- **Phase (1) - Compute public share values:**
  - *Client-side operations (executed by n_t clients in parallel)*: Send the public values corresponding to the client's private values to the master node.
  - *Master node operations*: Compute the public shared values and distribute them, along with the global model W^t, to all other clients participating in the round.
- **Phase (2) - Secure Sum Computation:**
  - *Client-side operations (executed by n_t clients in parallel)*:
    - ∗ Locally train the model W^t on its data D_i over E epochs, yielding W_i^{t+1}.
    - ∗ Compute mask values for W_i^{t+1} using the corresponding client i private keys.
    - ∗ Transmit the masked local model $Mask(W_i^{t+1})$ to U_{master}.
  - *Master node operations*:
    - ∗ Locally train the model W^t on its data D_{master} over E epochs, producing W_{master}^{t+1}.
    - ∗ Combine the received masked models and its own model, and execute the secure sum computation phase to obtain the global model:

$$W^{t+1} \leftarrow \sum_{i=1}^{n_t} \frac{m_i}{m} W_i^{t+1} + \frac{m_{\text{master}}}{m} W_{\text{master}}^{t+1}.$$

    - ∗ Transmit the global model W^{t+1} to all clients.

3.2.2. Experimental setup

Experiments were conducted using the MNIST and UCI SMS Spam datasets to assess the proposed framework. This section presents the efficiency of the pro-

Table 3.6: Size of the training and test datasets

	MNIST	SMS SPAM
Training size	60,000	4,457
Testing size	10,000	1,115

posed framework in handling IID, Non-IID, and imbalanced data with a low cost of communication bandwidth and high accuracy. The remaining part of this section is structured as follows: a description of the experiment, the experimental results, and the evaluation.

3.2.2.1. Datasets

The evaluation of the proposed framework involves two major datasets commonly used as benchmarks in deep learning research.

- **MNIST Dataset:** The first dataset is the MNIST dataset [189], which consists of images of handwritten digits. Each digit is formatted as a 32x32 image, normalized, centered, and resized to 28x28 pixels. The dataset contains 60,000 training examples and 10,000 test examples, making it a standard benchmark for image classification tasks in machine learning.
- **UCI SMS Spam Collection:** The second dataset is the UCI SMS Spam Collection [190], an imbalanced dataset comprising SMS messages labeled as either ham (legitimate) or spam. The dataset includes a total of 5,572 messages, of which 746 are spam and 4,826 are ham. For the purpose of this evaluation, the dataset was randomly split into 80% for training and 20% for testing, resulting in 4,457 training samples and 1,115 test samples.

Table 3.6 provides a summary of the number of training and test samples used in the evaluation.

3.2.2.2. Data Partitioning Strategies

To evaluate the performance of the proposed framework, two data partitioning methods are explored: IID (Independent and Identically Distributed) and Non-IID

(Non-Independent and Identically Distributed).

The IID approach ensures that the data is randomly shuffled and evenly distributed across all clients, providing each client with a representative subset of the entire dataset. For example, when applying the IID method to the MNIST dataset, if the data is partitioned among 100 clients, each client would receive an equal share of 600 examples, ensuring a balanced mix of digit images across all clients. Similarly, with the UCI SMS Spam Collection, the IID strategy would distribute both spam and ham messages equally among the 40 clients, maintaining a uniform distribution.

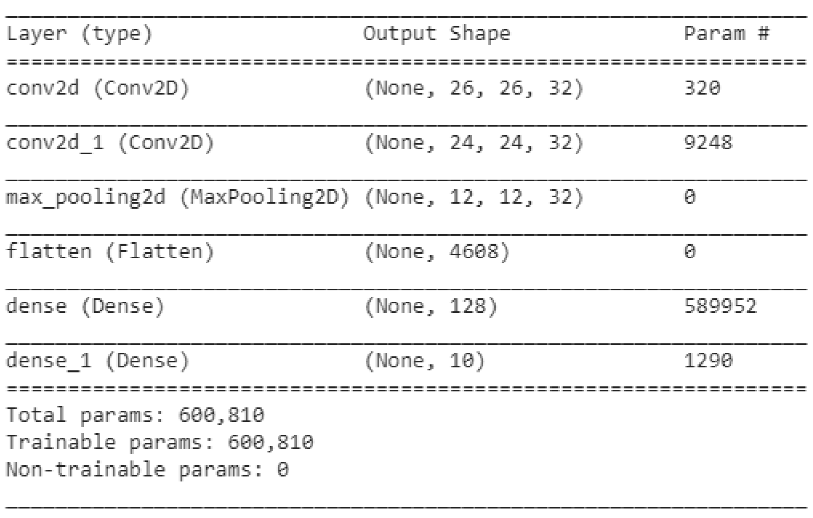
Conversely, the Non-IID approach reflects more realistic scenarios where data may not be evenly distributed across clients. In this method, the data is partitioned based on specific characteristics, such as class labels or message types, leading to imbalanced distributions. For instance, in the case of the MNIST dataset, the Non-IID partitioning might involve sorting the data by digit labels and dividing it into 200 shards, each containing 300 examples. Each of the 100 clients would then receive two shards, resulting in some clients having a higher concentration of certain digit images than others. Similarly, for the UCI SMS Spam Collection, applying a Non-IID partitioning strategy would mean some clients receive predominantly spam messages, while others receive mostly ham messages, thereby creating a varied distribution of data that mirrors real-world imbalances.

By employing both IID and Non-IID partitioning strategies, we can thoroughly evaluate the robustness and effectiveness of the proposed framework across different data distribution scenarios, from balanced to highly skewed distributions. This comprehensive approach allows us to better understand the framework's performance under diverse and realistic conditions.

3.2.2.3. Neural Network Architectures

In this experiment, two distinct neural network architectures were utilized: a Convolutional Neural Network (CNN) for the MNIST dataset and a Long Short-Term Memory (LSTM) network for the SMS spam detection dataset. CNNs are particularly well-suited for image recognition tasks due to their ability to capture spatial hierarchies in data through layers with sparse connectivity [191]. LSTMs, a type of Recurrent Neural Network (RNN), are designed to effectively model sequential dependencies, making them ideal for tasks involving time series or text data [192].

For the MNIST dataset, the network takes input images of handwritten digits, each represented as a 28x28 pixel grid, flattened into a single vector of 784 units. The network's goal is to accurately classify these inputs into one of ten possible digit categories, which defines the output layer with 10 units.

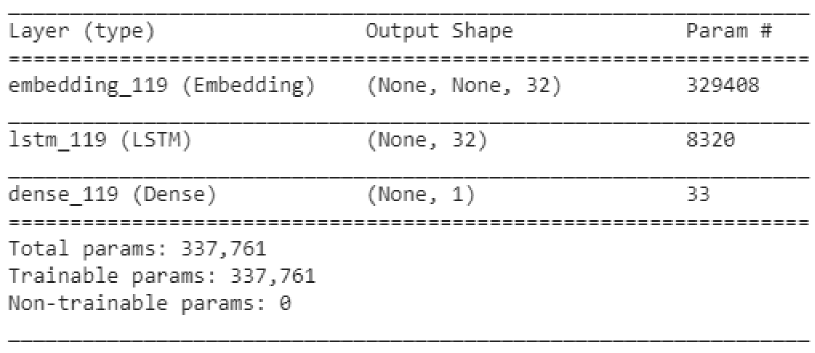


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 128)               589952
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 600,810
Trainable params: 600,810
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.10: CNN Model for MNIST dataset Summary.

In the case of the SMS Spam dataset, the input size is determined by the use of the top 10,000 most frequent words as features, resulting in an input vector of size 10,000. The classification task is to distinguish between spam and ham (non-spam) messages, necessitating an output layer with 2 units to represent these two classes.

Figures 3.10 (for CNN) and 3.11 (for LSTM) provide a visual summary of the network architectures, implemented using the Keras library. The CNN model contains 600,810 parameters, optimized for the image classification task, while the LSTM model, with 337,761 parameters, is tailored to handle the sequential nature of text data.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_119 (Embedding)    (None, None, 32)          329408
_________________________________________________________________
lstm_119 (LSTM)              (None, 32)                8320
_________________________________________________________________
dense_119 (Dense)            (None, 1)                 33
=================================================================
Total params: 337,761
Trainable params: 337,761
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.11: LSTM Model Summary.

3.2.2.4. Computing Framework

The training of the proposed framework was conducted on a desktop computer equipped with an Intel i3-9100F CPU, 16GB of RAM, and a GTX1060 GPU, all running on the Windows 10 Pro operating system. This setup provided a balanced combination of processing power and graphical capability suitable for deep learning tasks.

The software stack included Anaconda version 4.7.12 for environment management, TensorFlow version 1.15.0 for backend computations, and Keras version 2.2.5 as the primary high-level API for neural network development. Keras [193] is widely recognized for its intuitive interface and modular design, which facilitates rapid prototyping and experimentation. It integrates seamlessly with TensorFlow, CNTK, and Theano, allowing researchers to leverage the strengths of these powerful backend engines. Keras supports an extensive array of neural network models, including both convolutional networks (CNNs) and recurrent networks (RNNs), and can easily combine these models to tackle complex tasks.

Moreover, Keras is optimized for both CPU and GPU performance, enabling efficient training and inference across different hardware setups. Its compatibility with nearly all Python versions from 2.7 to 3.6 further enhances its flexibility and accessibility, making it an ideal choice for diverse research environments and experimental needs.

3.2.3. Experimental results and evaluation

3.2.3.1. Model performance

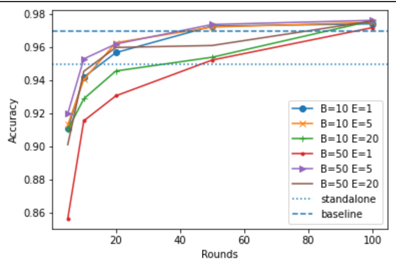
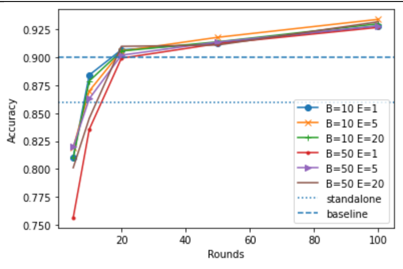
Overall model performance. The performance of the Secure Decentralized Training Framework (SDTF) is illustrated in Table 3.7. After 100 communication rounds, the proposed framework archived an accuracy of 97.6% on IID dataset, while the figure for Non-IID dataset was 93%. Despite the changes in the local parameters, the framework can be seen as a high-performance, and high accuracy model.

In this experiment, $K = 100$ is chosen as the full node for joint model training to showcase the performance of the proposed framework. The evaluation is conducted by varying the local batch size and local training epochs. Considering resource constraints, two batch sizes are considered: 10 and 50. Furthermore, local training epochs of 1, 5, and 20 are selected sequentially across 100 communication rounds. The impact on the model's accuracy is assessed at five checkpoints: the 5th, 10th, 20th, 50th, and 100th rounds.

With a local batch size of 50 and 1 local training epoch, the accuracy for both IID and non-IID data starts at 85% and 76% in the 5th communication round. After 20 rounds of training, a significant change in accuracy is observed for both the IID and non-IID MNIST datasets.

In the IID partition, the accuracy of the global model at a certain round is reported, considering a baseline accuracy of 97%. This baseline is chosen as it is significantly higher than the accuracy achievable through standalone local training (i.e., 95%). It is observed that with increased computation per client in each round, the communication costs to reach the baseline decrease. In all cases, the baseline accuracy of 97% is achieved in less than 100 communication rounds. However, scenarios with small batch sizes, large local epochs, and large batch sizes with small local epochs may lead to slower achievement. Linear interpolation between discrete points is used to compute the number of rounds needed for the model to achieve the target accuracy. As detailed in the next section, the framework achieves the 97% baseline approximately five times faster than Downpour SGD.

Table 3.7: The results on accuracy for different local training parameters on MNIST CNN (IID and Non-IID)

MNIST CNN IID	MNIST CNN Non-IID
	

After 100 rounds, the achieved accuracy baseline for non-IID partitioned data is 92% for all cases, instead of 97% as in the IID partition. However, in this case, local hyperparameters have less impact on model accuracy as the communication rounds increase. Although the performance in this scenario is inferior to the IID partition scenario, it is still better than standalone training, highlighting the robustness of the framework.

The next experiment compares the accuracy of the proposed framework and several training strategies.

Model accuracy comparison. The second experiment compares the proposed framework's accuracy with six other training strategies for a CNN model during the first 100 communication rounds. The CNN standalone version represents the training process in which each party uses only their local dataset for training, and the CNN centralized version involves storing and training the entire dataset centrally. These serve as two baseline cases for the experiment. The objective is to demonstrate that the framework achieves significantly higher model accuracy than standalone training. The comparison includes Selective learning and FedAVG with different hyperparameters. Two cases of 10% and 50% model parameter sharing are considered for Selective learning. With FedAVG, the framework is compared with its large and small noise versions. These strategies are chosen as they are known to be efficient in practice. Other cryptographic techniques may achieve accuracy similar to the model, but they are unsuitable for the semi-honest model and have high communication and computation costs. The cooperation results are presented in Table 3.8.

Table 3.8: Model accuracy comparison: balanced dataset

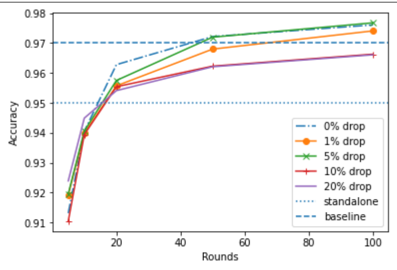
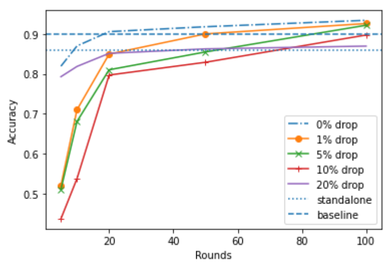
	Selective 10%	Selective 50%	FedAVG		Downpour SGD	CNN Centralize	CNN Standalone	SDTF
			Large Noise	Small Noise				
5	0.7436	0.8141	0.75	0.899	0.8913	0.9756	0.9061	0.9616
10	0.7902	0.8417	0.802	0.901	0.9295	0.9824	0.9341	0.9798
20	0.8171	0.8686	0.866	0.934	0.9518	0.9889	0.9354	0.9803
50	0.8214	0.8991	0.871	0.945	0.9768	0.9901	0.9355	0.9843
100	0.8862	0.9105	0.88	0.96	0.9817	0.9912	0.9479	0.9857

The CNN Centralize model clearly outperformed others, with the highest accuracy of 99.12% after 100 rounds. Centralizing the entire dataset for training can achieve higher accuracy levels than other approaches. However, this comes at the cost of privacy, as the whole dataset is centralized. To maintain privacy, selective learning and FedAVG with noise are employed. These models exhibited lower accuracy, even below the CNN Standalone model, in the first 100 rounds. Selective learning experiences a significant reduction in accuracy due to the loss of information in parameter sharing, leading to slower convergence and lower accuracy in the initial rounds. This behaviour is similar to Downpour SGD, which also shows slower convergence. Notably, the proposed framework with a batch size 32 and local epochs of 10 achieves the 97% baseline five times faster than Downpour SGD. FedAVG with noise, using a differential privacy approach, results in a substantial reduction in model accuracy, especially in the case of the large noise version. Selective learning and Downpour SGD also send raw parameters to other parties, compromising security.

The proposed framework performed better than most of the provided CNN model in this scenario with the accuracy of 98.57% after 100 communication rounds.

Impact of Dropped Nodes Ratio on Performance. In this section, the proposed framework's performance is evaluated in the presence of node dropout, considering varied dropout ratios. The batch size is fixed at 5, and the number of local epochs is set to 10. The evaluation is conducted at five checkpoints: the 5th, 10th, 20th, 50th, and 100th rounds. The dropout ratios investigated include 1%, 5%, 10%,

Table 3.9: Results on the framework's performance for the dropped nodes on the MNIST dataset. The proposed framework's performance have been tested in four scenarios of dropping 1%, 5%, 10% and 20% number of nodes through 5, 10, 20, 50 and 100 communication rounds

MNIST CNN IID	MNIST CNN Non-IID
	

and 20%, corresponding to 99%, 95%, 90%, and 80% of nodes participating in each round, respectively. The results are summarized in Table 3.9.

The model's performance improves with an increase in communication rounds. In the IID case, a small dropout ratio has a minor impact on performance, while larger dropout ratios (above 10%) lead to decreased performance. However, after a sufficiently large number of training rounds, the performance increases and approaches that of the case with no dropout. In all cases, the model's accuracy surpasses the baseline of the standalone model, set at 95%.

The model's performance improves with increased communication rounds for the Non-IID case. However, due to the impact of heterogeneous data, there is a decrease in performance compared to the IID case. Nevertheless, it remains significantly higher than standalone training.

The experiment highlights that the number of nodes dropped out is insignificant in the performance of the proposed framework in the IID data case. However, in the Non-IID data case, it can incur additional costs and affect convergence. Nonetheless, the framework achieves baseline performance with a sufficiently large number of communication rounds. The dropout ratio primarily influences the model's convergence, necessitating more training rounds to achieve satisfactory performance. The proposed framework demonstrates resilience in network volatility, requiring only a

Table 3.10: Model accuracy comparison: imbalanced dataset

	Selective 10%	Selective 50%	Downpour SGD	FedAVG (DP)	LSTM Centralize	LSTM standalone	SDTF
5	0.9499	0.9507	0.9634	0.9568	0.9676	0.8645	0.9658
10	0.9551	0.9542	0.9641	0.9563	0.9689	0.867	0.9677
20	0.9559	0.9561	0.9684	0.9683	0.9782	0.9052	0.9686
50	0.9623	0.9678	0.9719	0.9696	0.9788	0.9134	0.9695
100	0.9688	0.9692	0.9726	0.9719	0.9813	0.9257	0.9721

certain number of nodes to participate in each round and allowing others to leave the network without compromising model accuracy.

Imbalanced datasets. The final experiment evaluates how the proposed framework handles imbalanced data, specifically using the UCI Spam Collection dataset with an 80:20 training-to-testing sets ratio. The Long Short-Term Memory (LSTM) model is employed for this scenario.

A comparison is drawn between the Secure Decentralized Training Framework (SDTF) and five different training strategies: selective learning with 10%, selective learning with 50%, DownpourSGD, federated learning, and LSTM centralized. The results are presented in Table 3.10.

In this experiment, the accuracy of these models is assessed over 100 communication rounds, with specific evaluation points at the 5th, 10th, 20th, 50th, and 100th rounds. The proposed model achieves an accuracy of 97.21% by the 100th round, outperforming most of the provided models. While the LSTM Centralized model retains the highest accuracy of 98.73% after 100 rounds, the proposed framework demonstrates remarkable accuracy in handling imbalanced data.

The experimental results also show that the proposed framework work well not only in IID data distribution but also in Non. IID data distribution. The model can obtain the 97% baseline of accuracy after only 10 rounds of communication in IID setting, that is 5× faster than Downpour SGD. The accuracy of the model in MNIST Non.IID settings and unbalance data with UCI SMS Spam dataset is much higher than any standalone training version. All these results can be obtained with only 10% of parties joint in each training rounds. This show that the framework is efficient even if many parties drop out training. So that it is very suitable for heterogeneous systems in which each party can drop out at any training round.

The system aims to address several privacy threats associated with deep learning. Next, the privacy of the framework is analyzed, demonstrating its capability to secure inputs, outputs, and local models from both direct and indirect leakage. Furthermore, a comparative assessment of the framework in terms of security, implementation performance, and utility is conducted using three approaches from the literature: the data-sharing approach, the ensemble method, and the model-sharing approach.

3.2.3.2. Privacy

In traditional distributed deep learning framework, the data owners and learning party are separate from each other. The data owners hold their local data and the learning party collects data from the owners to perform a deep model training process. Data owners neither control over the learning objective nor access the trained model directly. They need to contribute their data without any controls over it to a third party who acts as a learning server. In this case, privacy threats occur in both training phase and using phase.

In training phase, the local training data from the data owners may reveal directly to the learning server itself, to attackers who compromised the server's data storage, and to law enforcement and intelligence outfits due to legal and extra-legal means.

In using phase, when the data owners want to use the trained model, they must send their inputs to the model owner who often is the learning party itself. This may expose not only their local data but also the output results.

This section demonstrates the Secure Decentralized Training Framework for deep learning models to address various privacy threats effectively. The framework ensures the privacy of input data in both the training and usage phases, safeguards the output results during the usage phase, and protects all local intermediate models throughout the training process.

As mention above, the thesis consider the semi-honest model in which all parties are not actively malicious and follow the protocol exactly but may attempt to infer sensitive information from other parties' data.

• **Direct leakage**

In training phase. In the framework, instead of sharing local training data, all parties share their intermediate model parameters with noise. So that the training data are stored locally and do not reveal to any one. The data owners have full permissions with their data itself, thus ensures strong privacy of the model.

In using phase. After training process of the framework, all participants can get a collaborative global model and so that can use it locally and privately without sending any more data to the network. This makes both input data and predictive results private. There is absolutely no leakage in the using phase.

• **Indirect leakage**

Indirect leakage occurs when adversaries can collect intermediate model parameters from a honest party. The framework can eliminate the indirect leakage by using one of the three proposed protocols. The three proposed protocols can perform computing sum of all paramater vectors without revealing any other information even in the case of there are $n-2$ parties collude each other.

In conclusion, the proposed framework for deep learning models can protect the local privacy of all semi-honest parties from both direct and indirect leakage in training phase and using phase.

3.2.3.3. Computational complexity

In the proposed protocol, each round involves the utilization of a master node for both parameter aggregation and training a version of the local master server before aggregation. Furthermore, during any training round, the necessity for joint participation is limited to K parties instead of all N parties, significantly reducing computational and communication costs within the protocol. Additionally, the protocols operate with floating-point real numbers of at most 64-bit precision, significantly smaller than any homomorphic encryption scheme, thereby markedly decreasing the implementation cost.

Given that time complexity is a crucial aspect of a training framework, it is denoted:

• $T_t^{(i)}$ is the time for party i get the model updated at training round t.

• $T_{train}^{(i)}$ is the time required for i-th party training his local model on his local data.

• $T_{enc}^{(i)}$ is the time for computing shared parameters by Secure Vector Sum Protocol.

• $T_{upload}^{(i)}$ is the time for uploading the encrypted models.

• $T_{download}^{(i)}$ is the time for downloading the global model from the master node.

• $T_{master}^{(i)}$ is the time for master node compute the final global model at each round.

• *nRound* is the total number of training rounds.

The time complexity of the system for a party i is estimated using the following formula:

$$T_t^{(i)} = \max_{j=1}^{K}\{T_{train}^{(j)} + T_{enc}^{(j)} + T_{upload}^{(j)}\} + T_{download}^{(i)} + T_{master} \tag{3.2.3}$$

In a training round t, K parties parallelly train their local models, and the total time required for all of them to complete the protocol is given by $\max_{j=1}^{K} T^{(j)}train + T^{(j)}enc + T_{upload}^{(j)}$. Subsequently, the master node computes and returns the global model to all parties. Thus, the time complexity for a party i to fully perform a training round t is computed using Equation 3.2.3. The total cost of the framework is then:

$$T_{total} = \left[\max_{j=1}^{K}\{T_{train}^{(j)} + T_{enc}^{(j)} + T_{upload}^{(j)}\} + \max_{i=1}^{N} T_{download}^{(i)} + T_{master}\right] nRound. \tag{3.2.4}$$

Among the operations, the computational cost of the local training operation $T^{(i)}_{train}$ is the most expensive. But it is the imperative step in all the research of model sharing approach in literature. In the protocol, the only extend time is the time for computing shared parameters $T^{(i)}_{enc}$ but it is much less than training time and can be negligible.

3.2.3.4. Comparing the framework with other strategies

Employing model sharing, as in federated learning [3], instead of a data sharing approach, such as in [122, 123], offers advantages in terms of accuracy measures. Unlike data sharing approaches, model sharing eliminates the need for estimating techniques and transforming methods to accommodate data encryption algorithms. In many applications in the big data era, the data volume can be enormous, making sharing impractical. Additionally, data sharing typically requires a central server to collect and train the model, introducing potential issues like single points of failure or bottleneck problems.

PATE [137] requires most of the teacher parties have high quality local model. So it enquires each party need to have many labelled data. And further, it do not work well with statistical heterogeneity system. The combination of PATE and differential privacy also make reduce in accuracy of the global model.

With cryptography, the model can protect the privacy of all sharing model during training process without accuracy reduction. Moreover, the proposed secure sum protocol can reduce the communication and computation cost of cryptography approach. This protocol also can ensure the privacy of all local model even if there are $n-2$ parties collude with each other that is not able to obtain in tradition cryptography approach [162,176]. A comparison of the proposed framework with other approaches is shown detail in Table 3.11.

The proposed framework address the trade-off among users privacy, implementation cost, and model utility by combination of Federated Averaging techniques and the secure decentralized training framework. It not only ensures the privacy of local parties but also retains the accuracy of the original Federated Averaging on De-

Models	Bottleneck	Privacy	Collusion	Utility Reduction	Heterogeneity Data	Heterogeneity System	Performance cost	Training Latency
Traditional Deep Learning	YES	-	-	No	YES	High	Low	-
Downpour SGD	YES	Low	-	No	YES	Low	Low	Medium
Data Sharing with HE	YES	High	YES	High	YES	High	High	Medium
Data Sharing with DP	YES	Medium	NO	High	YES	High	Low	Low
Ensemble Learning PATE	YES	Medium	NO	Medium	NO	Low	Low	Low
Selective learning	YES	Low	-	Low	YES	Low	Low	High
Selective learning + HE	YES	High	YES	High	YES	Low	Medium	High
FedAVG	YES	Low	-	Low	YES	High	Low	Low
FedAVG + HE	YES	High	YES	Low	YES	High	Medium	Medium
FedAVG + DP	YES	Medium	NO	High	YES	High	Low	Low
Blockchain base	NO	High	YES	High	YES	High	High	Medium
Proposed Framework	NO	High	NO	Low	YES	High	Low	Low

Table 3.11: Comparison among privacy preserving deep learning approaches

centralize network without any third party server. Moreover, the model also work well with the systems and statistical heterogeneity in a high performance.

3.3. Chapter Summary

In this thesis section, the application of secure multiparty computation protocols, as detailed in Chapter 2, has been explored for training distributed deep learning models, encompassing both centralized and decentralized approaches. The results highlight the effectiveness of the mentioned protocols in various settings, covering diverse datasets and deep neural network models. The research findings presented in this chapter have been documented in the publications [2,3,4,5,6,7].

CONCLUSION AND FUTURE WORK

Deep learning has surfaced as an immensely potent methodology across diverse machine learning domains, encompassing, among others, image classification, speech recognition, natural language processing (NLP), and bioinformatics. The effectiveness of deep learning approaches is inherently connected to the amount of data available for training purposes. There is a significant interest among multiple entities to train a global model using their respective datasets collaboratively. Nevertheless, such data collection frequently involves sensitive user information, giving rise to several privacy concerns. In order to address this issue, Google Brain has implemented federated learning, which allows for the storage of training data on local devices and the learning of a shared model through the aggregation of locally calculated updates.

In order to use the Federated learning technique, participants are obligated to engage in the process of parameter vector averaging. The act of sharing parameters in this procedure may inadvertently lead to the unintended disclosure of data, compromising the participants' privacy. The current body of research primarily focuses on addressing this matter through integrating methodologies such as differential privacy, which involves the introduction of random noise, or cryptographic-based approaches like SMC, to augment the security of the parameter-sharing procedure. Nevertheless, despite the numerous advancements made in cryptographic approaches and differential privacy, there is still a need to enhance the security and efficiency of exchanging local models.

Although differential privacy methods frequently result in reduced model accuracy, necessitating a trade-off between privacy and efficiency, cryptographic-based approaches, specifically secure multiparty computation protocols, offer promising solutions for enhancing both security and efficiency. However, considering the existing constraints, the present scenario requires additional study endeavors that can proficiently utilize cryptographic techniques to attain practical efficacy. One of the disadvantages of this approach is the requirement to convert floating-point real numbers into big integers, which is a computationally demanding procedure. This conversion introduces a loss of accuracy during the training of the model. Furthermore, implementing a protocol is essential to guarantee the utmost level of security in situations where collusion poses a substantial obstacle. This thesis focuses on the proposition and enhancement of secure multiparty computation protocols to address and overcome the aforementioned shortcomings.

This thesis presents three multiparty secure sum techniques as potential solutions to the aforementioned problems. The security aspect and performance of each idea were investigated and evaluated in the thesis. The findings of the thesis can be succinctly described in the following manner:

- The thesis has proposed three secure multi-party sum protocols that work well on floating-point real number vectors in semi-honest collision scenarios. The first protocol used a method that segregated the integer and decimal components and a modified version of the ElGamal cryptographic system. The fractional component is compressed using varied levels of precision. This protocol ensures the preservation of differential privacy by utilizing the inherent randomness in estimation errors and attains computational security by employing cryptographic methods. The federated learning protocol guarantees the learned model's precision by utilizing an appropriate compression ratio. Nevertheless, it is necessary to meticulously select the compression ratios for each particular problem inside the experiment in order to uphold the intended level of precision. Moreover, the execution of two cryptographic computation phases results in a substantial increase in both the computing and communication expenses associated with this protocol. Hence, this thesis presents an alternative protocol that utilizes the masking matrix approach in conjunction with a modified version of the ECC cryptosystem. The objective of this protocol is to attain a significant level of precision while concurrently minimizing the expenses associated with communication and processing. The efficacy of the two proposed protocols is seen in their ability to achieve high levels of efficiency when applied to diverse datasets and neural network topologies. The attained level of accuracy demonstrates just a slight decline compared to conventional federated learning methods. Nevertheless, the two methods lack data authentication and are vulnerable to membership spoofing attacks. Therefore, the thesis presents a novel protocol as a potential solution to tackle this particular problem. The proposed protocol utilizes a composite approach consisting of a random noise masking matrix, different iterations of the Elgamal cryptosystem, hashing methods, and digital signatures in order to guarantee the secrecy, privacy, and validity of the data belonging to the involved parties. The proposed secure sum protocols consist of three unique approaches that facilitate the collective computation of a sum of private inputs by a substantial group of n participants while operating under the assumption of semi-honest behavior. These protocols are designed to withstand collusion among $n-2$ parties. The aforementioned protocols demonstrate efficacy when applied to floating point numbers, rendering them highly compatible with the implementation of federated learning.

- The thesis proposes the utilization of these protocols for training federated learning models in both centralized and decentralized network contexts, with the aim of assessing their effectiveness. The thesis undertakes a theoretical evaluation of the privacy and communication expenses linked to the training procedure. The research entails the implementation of empirical examinations on multiple datasets, namely the MNIST dataset, characterized by its equitable distribution of class images; the SMS Spam dataset, recognized for its imbalanced class text data; and the CSIC 2010 dataset, which specifically concentrates on the identification of web attacks. This research employs various deep neural network architectures, including Convolutional Neural Networks (CNN), Convolutional Long Short-Term Memory (CLCNN), and Long Short-Term Memory (LSTM). The experiments conducted as part of this study provide empirical evidence that supports the claim that the proposed methodology has the potential to attain a significant level of accuracy. For example, the model attained a baseline accuracy of 97% following ten training iterations using the MNIST dataset and fifty training iterations using the SMS Spam dataset. Moreover, the methodology shows robustness in diverse, distributed networks characterized by non-identically and independently distributed (non-IID) and imbalanced data distributions. Moreover, empirical evidence demonstrates a fivefold reduction in the required number of training iterations to achieve the accuracy baseline compared to Downpour SGD.

The proposed methodology demonstrates the ability to attain robust privacy at the cryptographic level while preserving a higher level of model utility than differential privacy methodologies. Including this feature enhances the efficiency and practicality of the proposed protocols, rendering them suitable for implementation in real-world scenarios. The protocols offered are novel, as they are designed to operate with floating-point real numbers. Their primary objective is to safeguard local models from any party who may possess honest intentions but is curious. This protection is ensured even when collusion occurs among $n-2$ out of n parties, specifically within the context of privacy-preserving deep learning.

Subsequently, the thesis examines prospective challenges that may arise within the broader domain of SMC in the forthcoming period.

- The genesis of the field of SMC may be traced back to distributed computing scenarios and practical challenges, as discussed in Chapter 2. Therefore, it is imperative for the research community to explore new secure multi-party computation protocols in response to evolving distributed computing situations and the demands of real challenges.
- Furthermore, it is imperative that secure multi-party computation protocols are founded upon contemporary cryptographic methods, such as post-quantum cryptographic approaches. This is crucial in order to safeguard against potential threats posed by upcoming advancements in computing technology that are rapidly approaching.
- The application of SMC can be integrated into the input sharing approach and ensemble learning for greater efficiency, as well as to solve some training model cases under certain conditions.

BIBLIOGRAPHY

[1] Mehdi Gheisari, Fereshteh Ebrahimzadeh, Mohamadtaghi Rahimi, Mahdieh Moazzamigodarzi, Yang Liu, Pijush Kanti Dutta Pramanik, Mohammad Ali Heravi, Abolfazl Mehbodniya, Mustafa Ghaderzadeh, Mohammad Reza Feylizadeh, et al. Deep learning: Applications, architectures, models, tools, and frameworks: A comprehensive survey. *CAAI Transactions on Intelligence Technology*, 8(3):581–606, 2023.

[2] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *ACM Computing Surveys*, 56(4):1–34, 2023.

[3] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

[4] H. McMahan, Eider Moore, D. Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *ArXiv*, abs/1602.05629, 2016.

[5] Mark Vero, Mislav Balunović, Dimitar Iliev Dimitrov, and Martin Vechev. Tableak: Tabular data leakage in federated learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 35051–35083. PMLR, 2023.

[6] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems*, 2022.

[7] Ahmed El Ouadrhiri and Ahmed Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE access*, 10:22359–22380, 2022.

[8] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[9] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[10] Guozhu Dong and Huan Liu. *Feature engineering for machine learning and data analytics*. CRC Press, 2018.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[12] Honglak Lee, Roger B. Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 609–616. ACM, 2009.

[13] Ruo-Yu Sun. Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 8(2):249–294, 2020.

[14] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 661–670, New York, NY, USA, 2014. Association for Computing Machinery.

[15] S. Sun, Z. Cao, H. Zhu, and J. Zhao. A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681, 2020.

[16] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[17] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[18] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, San Diego, CA, August 2014. USENIX Association.

[19] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 587–601, New York, NY, USA, 2017. Association for Computing Machinery.

[20] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 1291–1308. USENIX Association, 2020.

[21] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 363–375, New York, NY, USA, 2020. Association for Computing Machinery.

[22] Zecheng He, Tianwei Zhang, and Ruby B. Lee. Model inversion attacks against

collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 148–162, New York, NY, USA, 2019. Association for Computing Machinery.

[23] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 603–618, New York, NY, USA, 2017. Association for Computing Machinery.

[24] Samuel Yeom, I. Giacomelli, Matt Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, 2018.

[25] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017.

[26] S. Truex, Ling Liu, M. Gursoy, L. Yu, and Wenqi Wei. Towards demystifying membership inference attacks. *ArXiv*, abs/1807.09173, 2018.

[27] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *ArXiv*, abs/1806.01246, 2019.

[28] Congzheng Song and Vitaly Shmatikov. The natural auditor: How to tell if someone used your words to train their model. *ArXiv*, abs/1811.00513, 2018.

[29] J. Hayes, Luca Melis, G. Danezis, and Emiliano De Cristofaro. Logan: Evaluating privacy leakage of generative models using generative adversarial networks. *ArXiv*, abs/1705.07663, 2017.

[30] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 601–618, USA, 2016. USENIX Association.

[31] B. Wang and N. Z. Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 36–52, 2018.

[32] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2003–2020. USENIX Association, August 2020.

[33] Seong Joon Oh, M. Augustin, M. Fritz, and B. Schiele. Towards reverse-engineering black-box neural networks. In *ICLR*, 2018.

[34] Nicolas Papernot, Patrick McDaniel, Ian 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*, ASIA CCS '17, page 506–519, New York, NY, USA, 2017. Association for Computing Machinery.

[35] M. Juuti, S. Szyller, S. Marchal, and N. Asokan. Prada: Protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 512–527, 2019.

[36] Nicolas Papernot, P. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *ArXiv*, abs/1605.07277, 2016.

[37] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4949–4958, 2019.

[38] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[39] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim. Privacy-preserving deep learning on machine learning as a service—a comprehensive survey. *IEEE Access*, 8:167425–167447, 2020.

[40] Pierangela Samarati and Sabrina De Capitani Di Vimercati. Data protection in outsourcing scenarios: Issues and directions. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 1–14, 2010.

[41] Souhaib Ben Taieb and Rob J Hyndman. A gradient boosting approach to the kaggle load forecasting competition. *International journal of forecasting*, 30(2):382–394, 2014.

[42] Blake Hallinan and Ted Striphas. Recommended for you: The netflix prize and the production of algorithmic culture. *New Media & Society*, 18(1):117–137, 2016.

[43] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[44] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[45] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[46] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008.

[47] Puneet Goswami and Suman Madan. Privacy preserving data publishing and data anonymization approaches: A review. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 139–142. IEEE, 2017.

[48] Wenliang Du and Zhijun Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proceedings of the ninth ACM SIGKDD*

international conference on Knowledge discovery and data mining, pages 505–510, 2003.

[49] Fan Wu, Jiqiang Liu, and Sheng Zhong. An efficient protocol for private and accurate mining of support counts. *Pattern recognition letters*, 30(1):80–86, 2009.

[50] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS ’14, page 1054–1067, New York, NY, USA, 2014. Association for Computing Machinery.

[51] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: Efficient and private neural network training. In *Privacy Enhancing Technologies Symposium.* (PETS 2019), February 2019.

[52] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Comput. Surv.*, 52(4), August 2019.

[53] Matthias Langer, Zhen He, Wenny Rahayu, and Yanbo Xue. Distributed training of deep learning models: A taxonomic perspective. *IEEE Transactions on Parallel and Distributed Systems*, 31(12):2802–2818, 2020.

[54] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), January 2019.

[55] D.B. Skillicorn and S.M. McConnell. Distributed prediction from vertically partitioned data. *Journal of Parallel and Distributed Computing*, 68(1):16–36, 2008. Parallel Techniques for Information Extraction.

[56] Jaideep Vaidya. Privacy preserving data mining over vertically partitioned data. *Purdue University, West Lafayette, IN*, 2004.

[57] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International*

Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05):557–570, 2002.

[58] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[59] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[60] Adi Shamir, Ronald L Rivest, and Leonard M Adleman. Mental poker. In *The mathematical gardner*, pages 37–43. Springer, 1981.

[61] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[62] Oded Goldreich. *Foundations of Cryptography, Volume 2*. Cambridge university press Cambridge, 2004.

[63] P. S. Naidu, R. Kharat, R. Tekade, P. Mendhe, and V. Magade. E-voting system using visual cryptography secure multi-party computation. In *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, pages 1–4, 2016.

[64] Peter Bogetoft, Ivan Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft. A practical implementation of secure auctions based on multiparty integer computation. In *International Conference on Financial Cryptography and Data Security*, pages 142–147. Springer, 2006.

[65] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631, 2017.

[66] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

[67] Ronald Cramer, Ivan Bjerre Damgård, et al. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[68] Manoj M Prabhakaran and Amit Sahai. *Secure multi-party computation*, volume 10. IOS press, 2013.

[69] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Found. Trends Priv. Secur.*, 2(2–3):70–246, December 2018.

[70] Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI global, 2005.

[71] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[72] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 277–346, 2017.

[73] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, June 1985.

[74] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 325–335, 2000.

[75] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 573–590, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[76] Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 284–302, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[77] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192, 2015. `https://eprint.iacr.org/2015/1192`.

[78] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), July 2018.

[79] Ciara Moore, Máire O'Neill, Elizabeth O'Sullivan, Yarkın Doröz, and Berk Sunar. Practical homomorphic encryption: A survey. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2792–2795. IEEE, 2014.

[80] Jing Yang, Mingyu Fan, Guangwei Wang, and Zhiyin Kong. Simulation study based on somewhat homomorphic encryption. *Journal of Computer and Communications*, 2(02):109–111, 2014.

[81] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[82] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[83] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, page 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[84] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of cryptography conference*, pages 325–341. Springer, 2005.

[85] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption

from (standard) lwe. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, 2011.

[86] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.

[87] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled fhe from learning with errors. In *Annual Cryptology Conference*, pages 630–656. Springer, 2015.

[88] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.

[89] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[90] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[91] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 315–335. Springer, 2013.

[92] Shai Halevi and Victor Shoup. Algorithms in helib. In *Annual Cryptology Conference*, pages 554–571. Springer, 2014.

[93] Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer, 2015.

[94] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[95] Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In Hillol Kargupta, Jaideep Srivastava, Chandrika Kamath, and Arnold Goodman, editors, *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*, pages 92–102. SIAM, 2005.

[96] F. Hao, P. Y. A. Ryan, and P. Zieliński. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, 2010.

[97] Feng Hao, Matthew N Kreeger, Brian Randell, Dylan Clarke, Siamak F Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in {Large-Scale} electronic voting. In *2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14)*, 2014.

[98] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On private scalar product computation for privacy-preserving data mining. In Choonsik Park and Seongtaek Chee, editors, *Information Security and Cryptology – ICISC 2004*, pages 104–120, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[99] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[100] Rashid Sheikh and Durgesh Kumar Mishra. Secure sum computation for insecure networks. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, ICTCS '16, New York, NY, USA, 2016. Association for Computing Machinery.

[101] Gilad Asharov and Yehuda Lindell. A full proof of the bgw protocol for perfectly secure multiparty computation. *J. Cryptology*, 30:58–151, 2017.

[102] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[103] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, page 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

[104] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[105] Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[106] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*, pages 114–121, 2009.

[107] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[108] Fatemehsadat Mirshghallah, Mohammadkazem Taram, Praneeth Vepakomma, Abhishek Singh, Ramesh Raskar, and Hadi Esmaeilzadeh. Privacy in deep learning: A survey. *arXiv preprint arXiv:2004.12254*, 2020.

[109] Joseph Geumlek and Kamalika Chaudhuri. Profile-based privacy for locally private computations. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 537–541. IEEE, 2019.

[110] Sivakanth Gopi, Pankaj Gulhane, Janardhan Kulkarni, Judy Hanwen Shen, Milad Shokouhi, and Sergey Yekhanin. Differentially private set union. *arXiv preprint arXiv:2002.09745*, 2020.

[111] Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning

differentially private recurrent language models. In *International Conference on Learning Representations (ICLR)*, 2018.

[112] Keke Chen and Ling Liu. Privacy preserving data classification with rotation perturbation. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[113] Linshan Jiang, Rui Tan, Xin Lou, and Guosheng Lin. On lightweight privacy-preserving collaborative learning for internet-of-things objects. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 70–81, 2019.

[114] Bin Liu, Yurong Jiang, Fei Sha, and Ramesh Govindan. Cloud-enabled privacy-preserving collaborative learning for mobile sensing. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 57–70, 2012.

[115] Yiran Shen, Chengwen Luo, Dan Yin, Hongkai Wen, Rus Daniela, and Wen Hu. Privacy-preserving sparse representation classification in cloud-enabled mobile applications. *Computer Networks*, 133:59–72, 2018.

[116] Kun Liu, Hillol Kargupta, and Jessica Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on knowledge and Data Engineering*, 18(1):92–106, 2005.

[117] Chong Huang, Peter Kairouz, Xiao Chen, Lalitha Sankar, and Ram Rajagopal. Context-aware generative adversarial privacy. *Entropy*, 19(12):656, 2017.

[118] Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. Differentially private mixture of generative neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1109–1121, 2018.

[119] Runhua Xu, Nathalie Baracaldo, and James Joshi. Privacy-preserving machine learning: Methods, challenges and directions. *arXiv preprint arXiv:2108.04417*, 2021.

[120] The Dung Luong and Dang Hung Tran. A new method of privacy preserving

computation over 2-part fully distributed data. In *The 9th International Conference on Computing and InformationTechnology (IC2IT2013) 9th-10th May 2013 King Mongkut's University of Technology North Bangkok*, pages 115–123. Springer, 2013.

[121] F. Bu, Y. Ma, Z. Chen, and H. Xu. Privacy preserving back-propagation based on bgv on cloud. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pages 1791–1795, 2015.

[122] J. Yuan and S. Yu. Privacy preserving back-propagation neural network learning made practical with cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):212–221, 2014.

[123] Qingchen Zhang, Laurence T. Yang, and Zhikui Chen. Privacy preserving deep computation model on cloud for big data feature learning. *IEEE Trans. Comput.*, 65(5):1351–1362, May 2016.

[124] Ping Li, Jin Li, Zhengan Huang, Tong Li, Chong-Zhi Gao, Siu-Ming Yiu, and Kai Chen. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, 74:76–85, 2017.

[125] 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, 2017.

[126] Lingjuan Lyu, Xuanli He, Yee Wei Law, and Marimuthu Palaniswami. Privacy-preserving collaborative deep learning with application to human activity recognition. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1219–1228, New York, NY, USA, 2017. Association for Computing Machinery.

[127] Charu C. Aggarwal and Philip S. Yu, editors. *Privacy-Preserving Data Mining - Models and Algorithms*, volume 34 of *Advances in Database Systems*. Springer, 2008.

[128] M. Al-Rubaie and J. M. Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security Privacy*, 17(2):49–58, 2019.

[129] Dayin Zhang, X. Chen, D. Wang, and Jinqiao Shi. A survey on collaborative deep learning and privacy-preserving. *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 652–658, 2018.

[130] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1655–1658, New York, NY, USA, 2018. Association for Computing Machinery.

[131] Pawel Rozycki, Janusz Kolbusz, Roman Korostenskyi, and Bogdan M Wilamowski. Estimation of deep neural networks capabilities using polynomial approach. In *International Conference on Artificial Intelligence and Soft Computing*, pages 136–147. Springer, 2016.

[132] H. Chabanne, Amaury de Wargny, J. Milgram, C. Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.*, 2017:35, 2017.

[133] Pengtao Xie, Misha Bilenko, Tom Finley, Ran Gilad-Bachrach, Kristin E. Lauter, and Michael Naehrig. Crypto-nets: Neural networks over encrypted data. *CoRR*, abs/1412.6181, 2014.

[134] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

[135] Aleksei Triastcyn and Boi Faltings. Generating differentially private datasets using gans. 2018.

[136] Nicolas Papernot, Martin Abadi, Ulfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*. OpenReview.net, 2017.

[137] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Ulfar Erlingsson. Scalable private learning with PATE. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[138] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. Learning privately from multiparty data. In *International Conference on Machine Learning*, pages 555–563, 2016.

[139] Franziska Boenisch, Christopher Mühl, Roy Rinberg, Jannis Ihrig, and Adam Dziedzic. Individualized pate: Differentially private machine learning with individual privacy guarantees. *arXiv preprint arXiv:2202.10517*, 2022.

[140] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

[141] Yunhui Long, Boxin Wang, Zhuolin Yang, Bhavya Kailkhura, Aston Zhang, Carl Gunter, and Bo Li. G-pate: Scalable differentially private data generator via private aggregation of teacher discriminators. *Advances in Neural Information Processing Systems*, 34:2965–2977, 2021.

[142] Chao-Han Huck Yang, Sabato Marco Siniscalchi, and Chin-Hui Lee. Pate-aae: Incorporating adversarial autoencoder into private aggregation of teacher ensembles for spoken command classification. *arXiv preprint arXiv:2104.01271*, 2021.

[143] Zhiliang Tian, Yingxiu Zhao, Ziyue Huang, Yu-Xiang Wang, Nevin L Zhang, and He He. Seqpate: Differentially private text generation via knowledge distillation. *Advances in Neural Information Processing Systems*, 35:11117–11130, 2022.

[144] Arnaud Grivet Sébert, Rafael Pinot, Martin Zuber, Cédric Gouy-Pailler, and Renaud Sirdey. Speed: secure, private, and efficient deep learning. *Machine Learning*, 110:675–694, 2021.

[145] Christopher A Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. Capc learning: Confidential and private collaborative learning. *arXiv preprint arXiv:2102.05188*, 2021.

[146] Andrew Hard, Chloé M Kiddon, Daniel Ramage, Francoise Beaufays, Hubert Eichner, Kanishka Rao, Rajiv Mathews, and Sean Augenstein. Federated learning for mobile keyboard prediction, 2018.

[147] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[148] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[149] Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, H. R. Rabiee, N. Lane, and H. Haddadi. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, 7:4505–4518, 2020.

[150] A. Ichinose, M. Oguchi, A. Takefusa, and H. Nakada. Evaluation of distributed processing of caffe framework using poor performance device. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3980–3982, 2016.

[151] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[152] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[153] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Commu-*

nications Security, CCS '15, page 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery.

[154] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[155] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[156] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[157] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[158] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[159] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A communication efficient collaborative learning framework for distributed features. *arXiv preprint arXiv:1912.11187*, 2019.

[160] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. A performance evaluation of federated learning algorithms. In *Proceedings of the second workshop on distributed infrastructures for deep learning*, pages 1–8, 2018.

[161] Seyedeh Maryam Hosseini, Milad Sikaroudi, Morteza Babaei, and Hamid R Tizhoosh. Cluster based secure multi-party computation in federated learning for histopathology images. In *International Workshop on Distributed, Collaborative, and Federated Learning*, pages 110–118. Springer, 2022.

[162] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[163] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[164] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019.

[165] Xiaojie Guo, Zheli Liu, Jin Li, Jiqiang Gao, Boyu Hou, Changyu Dong, and Thar Baker. V eri fl: Communication-efficient and fast verifiable aggregation for federated learning. *IEEE Transactions on Information Forensics and Security*, 16:1736–1751, 2020.

[166] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[167] Beongjun Choi, Jy-yong Sohn, Dong-Jun Han, and Jaekyun Moon. Communication-computation efficient secure aggregation for federated learning. *arXiv preprint arXiv:2012.05433*, 2020.

[168] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[169] Jinhyun So, Başak Güler, and A Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, 2021.

[170] Constance Beguier and Eric W Tramel. Safer: Sparse secure aggregation for federated learning. *arXiv preprint arXiv:2007.14861*, 2020.

[171] Anh Tu Tran, The Dung Luong, Van Nam Huynh, and Viet Hung Dang. A novel privacy preserving deep learning protocol for sms spam detection in distributed mobile environment. In *Proceedings of The 23rd National Symposium of Selected ICT Problems*, pages 108–114, 2020.

[172] Renuga Kanagavelu, Qingsong Wei, Zengxiang Li, Haibin Zhang, Juniarto Samsudin, Yechao Yang, Rick Siow Mong Goh, and Shangguang Wang. Ce-fed: Communication efficient multi-party computation enabled federated learning. *Array*, 15:100207, 2022.

[173] Anh Tu Tran, The Dung Luong, Van Nam Huynh, and Viet Hung Dang. Privacy preserving sms spam detection with deep learning models in distributed environment. In *Proceedings of the 13th National Conference on Fundamental and Applied Information Technology Research (FAIR'2020)*, pages 508–515, 2020.

[174] David Byrd and Antigoni Polychroniadou. Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–9, 2020.

[175] Vaikkunth Mugunthan, Antigoni Polychroniadou, David Byrd, and Tucker Hybinette Balch. Smpai: Secure multi-party computation for federated learning. In *Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services*, pages 1–9. MIT Press Cambridge, MA, USA, 2019.

[176] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *Trans. Info. For. Sec.*, 13(5):1333–1345, May 2018.

[177] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 1–11, 2019.

[178] Ye Dong, Xiaojun Chen, Liyan Shen, and Dakui Wang. Eastfly: Efficient and secure ternary federated learning. *Computers & Security*, 94:101824, 2020.

[179] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 493–506, 2020.

[180] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 13–23, 2019.

[181] Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. Poseidon: Privacy-preserving federated neural network learning. *arXiv preprint arXiv:2009.00349*, 2020.

[182] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 56–62. IEEE, 2021.

[183] Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. The past, evolving present, and future of the discrete logarithm. *Open Problems in Mathematics and Computational Science*, pages 5–36, 2014.

[184] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[185] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4:161–174, 1991.

[186] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[187] Daniel Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Math. Soc., 1971*, volume 20, pages 415–440, 1971.

[188] Annachiara Ruospo, Ernesto Sanchez, Marcello Traiola, Ian O'connor, and Alberto Bosio. Investigating data representation for efficient and reliable convolutional neural networks. *Microprocessors and Microsystems*, 86:104318, 2021.

[189] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[190] Tiago Almeida and Jos Hidalgo. SMS Spam Collection. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5CC84.

[191] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[192] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[193] Francois Chollet et al. Keras, 2015.

PUBLICATION LIST

RELATED PUBLICATIONS TO THE THESIS

1. **Anh-Tu Tran**, The-Dung Luong, Van-Nam Huynh. A Comprehensive Survey and Taxonomy on Privacy-Preserving Deep Learning. *Neurocomputing*, Volume 576, 2024, Pages 127345 (SCI/ISI indexed, Scopus Q1).

2. **Anh-Tu Tran**, The-Dung Luong, Jessada Karnjana, Van-Nam Huynh. An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing*, Volume 422, 2021, Pages 245-262, ISSN 0925-2312. (SCI/ISI indexed, Scopus Q1).

3. **Anh-Tu Tran**, The Dung Luong, and Xuan Sang Pham. A Novel Privacy-preserving Federated Learning Model based on Secure Multi-party Computation. *In: International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making. Cham: Springer Nature Switzerland*, 2023.

4. **Anh-Tu Tran**, and Xuan Sang Pham. A novel privacy-preserving deep learning scheme for the classification of Covid-19 in chest x-ray images. *The 15th IEEE International Conference on KNOWLEDGE AND SYSTEMS ENGINEERING (KSE 2023)*, 2023.

5. **Anh-Tu Tran**, Van Vu Thi, Xuan Sang Pham. A Novel Federated Learning Model with Integer Encoding Method for Web Attack Detection. In *Proceedings of the 15th National Conference on Fundamental and Applied Information Technology Research (FAIR'2022)*, 2022.

6. **Anh-Tu Tran**, The Dung Luong, and Xuan Sang Pham. Privacy-preserving Deep Learning Model with Integer Quantization and Secure Multi-party Computation. Annals of Operations Research, 2024 (SCI/ISI indexed, Scopus Q1) (Accepted).

7. **Anh-Tu Tran**, Van-Nam Huynh, Viet-Hung Dang. A Novel Privacy Preserving Framework for Training Dempster Shafer Theory-based Evidential Deep Neural

Network. *In International Conference on Belief Functions (pp. 98-107). Cham: Springer Nature Switzerland.* (ISI indexed, Scopus).

THE ADDITIONAL RELATED PUBLISHINGS

1. **Anh Tu Tran**, The Dung Luong, Viet Hung Dang, Van Nam Huynh. Using Anonymous Protocol for Privacy Preserving Deep Learning Model. In *Proceedings of 2020 7th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 60-65, IEEE, 2020.

2. **Anh-Tu Tran**, The-Dung Luong, Cong-Chieu Ha, Duc-Tho Hoang, Thi-Luong Tran. Secure Inference via Deep Learning as a Service without Privacy Leakage. In *Proceedings of 2021 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 1-6, IEEE, 2021.

3. **Anh-Tu Tran**, The Dung Luong, Xuan Sang Pham, Thi Luong Tran. Deep Models with Differential Privacy for Distributed Web Attack Detection. In *Proceedings of 2022 14th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1-6, IEEE, 2022.

4. **Tran Anh Tu**, Luong The Dzung, Hoang Duc Tho, Nguyen Hoang Anh. A novel secure deep ensemble learning protocol based on Conjugacy search problem homomorphic encryption scheme. *Journal of Science and Technology on Information security*, Special Issue CS(15), pages 7-16, 2022.

5. **Tran Anh Tu**, Luong The Dung, Huynh Van Nam, Dang Viet Hung. A Novel Privacy Preserving Deep Learning Protocol for SMS Spam Detection in Distributed Mobile Environment. In *Proceedings of The 23rd National Symposium of Selected ICT Problems*, pages 108-114, 2020.

6. **Tran Anh Tu**, Luong The Dung, Huynh Van Nam, Dang Viet Hung. Privacy Preserving SMS Spam Detection with Deep Learning Models in Distributed Environment. In *Proceedings of the 13th National Conference on Fundamental and Applied Information Technology Research (FAIR'2020)*, 2020.

7. Linh Trinh, Bach Ha, **Anh Tu Tran**. VQC-COVID-NET: Vector Quantization Contrastive Learning for Covid-19 Image Base Classification, In *Proceedings*

of 9th NAFOSTED Conference on Information and Computer Science (NICS), pages 247-251, 2022.

8. Linh Trinh, Bach Ha, **Anh Tu Tran**. FisheyePP4AV: A privacy-preserving method for autonomous vehicles on fisheye camera images. *2023 RIVF International Conference on Computing and Communication Technologies (RIVF).*

9. Vu Duy Hien, Luong The Dung, and **Tran Anh Tu**. A solution for privacy-preserving item-based recommendation system in fully distributed setting. In *Proceedings of Symposium on Information Security*, 2018.

10. **Anh-Tu Tran**, The-Dung Luong, Van-Nam Huynh. Privacy-Preserving Federated Learning Scheme with Colluding Parties Based on ECC-like Secure Multi-Party Computation Protocol. *Computer & Security* (SCI/ISI indexed, Scopus Q1) (Under Reviewed).

11. **Anh-Tu Tran**, The-Dung Luong, Van-Nam Huynh. Secure and Verifiable Aggregation Scheme for Federated Learning based on Secure Multi-party Computation. *Neurocomputing* (SCI/ISI indexed, Scopus Q1) (Under Reviewed).