MINISTRY OF EDUCATION
AND TRAINING

VIETNAM ACADEMY OF
SIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SIENCE AND TECHNOLOGY

TRAN ANH TU

DEVELOPING CRYPTOGRAPHY-BASED SOLUTIONS TO ENHANCE SECURITY IN FEDERATED LEARNING

SUMMARY OF DISSERTATION ON COMPUTER SCIENCE
Code: 9 48 01 01

Ha noi - 2024

The dissertation is completed at: Graduate University of Science and Technology, Vietnam Academy of Science and Technology

Supervisors:

Supervisor 1: Assoc. Prof. Dr. Luong The Dung, Academy of Cryptography Technologies

Supervisor 2: Prof. HUYNH VAN NAM, Japan Advanced Institute of Science and Technology

Referee 1: …

Referee 2: …

Referee 3: ….

The dissertation will be examined by Examination Board of Graduate University of Science and Technology, Vietnam Academy of Science and Technology at…….. (date: , year:)

This dissertation can be found at:

1) Graduate University of Science and Technology Library
2) National Library of Vietnam

CONTENTS

ABSTRACT

Problem Statement

Deep learning is a powerful tool for various machine learning tasks, relying heavily on large training datasets. However, concerns over privacy have led to the adoption of federated learning, which allows for collaborative model training while keeping data on local devices. Despite its benefits, federated learning faces challenges such as indirect data leakage, prompting research into privacy-preserving techniques like differential privacy and cryptographic methods. While these methods offer improved security, they may impact model accuracy and introduce computational complexities. This thesis seeks to enhance secure multiparty computation (SMC) protocols to address these challenges and improve the efficiency and security of federated learning in real-world scenarios.

Thesis Objectives

The research objectives of this thesis are as follows:

- Crafting robust and efficient SMC protocols to precisely compute the sum of real-number vectors within a semi-trusted environment susceptible to collusion.
- Proposing new distributed deep learning training frameworks that guarantee both accuracy and efficiency by seamlessly incorporating federated learning strategies and the novel proposed SMC protocols.

Main Contributions

The thesis makes the following contributions:

- The first contribution involves proposing three novel SMC protocols designed to accurately compute the sum of real-number vectors within a semi-trusted environment prone to collusion by $n-2$ out of n parties.
- The second contribution entails the development of new distributed deep learning training frameworks that ensure both accuracy and efficiency by seamlessly integrating federated learning strategies and proposed SMC protocols.

Thesis organization

The primary thesis content unfolds across three chapters, excluding the Abstract and Conclusion.

- Chapter 1: Introduction navigates the landscape of Privacy-Preserving Deep Learning, assessing three key approaches: input sharing, model sharing, and output sharing. The thesis focuses on resolving challenges in training distributed deep learning networks through a model-sharing paradigm, which involves sharing parameters from local models to improve a global model's accuracy while protecting local training data. However, sharing parameters poses vulnerabilities to attacks like membership inference and model inversion, leading to data leakage. To mitigate these risks, the thesis proposes a fusion of SMC techniques to compute real-number vectors sum, aiming to synthesize a global model without compromising data from shared local models.
- Building on a deep analysis of the limitations of certain SMC protocols when applied to federated learning (FL) problems in Chapter 1, this dissertation proposes three novel secure multi-party summation protocols for privacy-preserving federated learning. These protocols include secure multi-party summation with integer quantization, secure multi-party vector summation using a masking matrix, and secure multi-party summation without requiring pre-established secure and authenticated communication channels. These protocols enable efficient computation of the sum of real-valued vectors while safeguarding against collusion of up to $n-2$ participants, including the aggregation server.
- Chapter 3 assesses the implemented proposed SMC protocols within federated learning models, exploring their performance in centralized and decentralized network settings. It encompasses evaluation scenarios across three datasets: MNIST, SMS Spam, and CSIC2010. Moreover, it examines three network topologies – CNN, LSTM, and CLCNN – within these evaluation scenarios.

CHAPTER 1. PRIVACY PRESERVING DEEP LEARNING

This chapter reviewed the challenges of privacy-preserving deep learning and key approaches to address them. Federated Learning shows promise for safeguarding privacy in neural network training, while cryptographic techniques offer secure parameter sharing. However, cryptographic methods face two major limitations: risks of collusion from key sharing and difficulties in handling real numbers, leading to potential precision loss. The foundational work from Chapter 1 has been published in **Publication 1**.

1.1. Deep learning

Deep learning involves multiple layers of abstraction, from data preprocessing to designing deep neural networks for capturing complex patterns. However, this approach faces challenges in terms of large data requirements and the high computational power needed for training.

1.2. Privacy-preserving Deep Learning

The efficiency of Deep Neural Networks heavily relies on the size of training datasets. Collaborative training of a global model faces a major hurdle: sharing local data among all parties. In response to privacy issues in collaborative deep learning, the concept of privacy-preserving deep learning has emerged [1].

1.3. Privacy Primitives

1.3.1. Anonymization

To protect privacy during model training, data is separated from its owner's identity, but simple anonymization (e.g., removing names) is often insufficient, as demonstrated by the Netflix Prize case.

1.3.2. Cryptographic techniques and Secure Multiparty Computation

1.3.2.1. Basic Concept

Definition 1.3.1. *Let K ($K \geq 2$) denote the cardinality of the set of members participating in the distributed computing network. Each member $i \in$*

$\{1,2,\ldots,K\}$ possesses an input $x_i \in X_i$. The function f is defined as a multi-party computing function as follows:

$$\begin{aligned} f &: X \to Y \\ \bar{x} = (x_1, x_2, \ldots, x_K) &\mapsto f(\bar{x}) = (f_1(\bar{x}), f_2(\bar{x}), \ldots, f_K(\bar{x})) \end{aligned} \tag{1.3.1}$$

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

1.3.2.2. Threat models

In Secure Multi-Party Computation (SMC), adversarial attacks are classified by behavior, power, and corruption type. Adversaries are either semi-honest or malicious based on their behavior, and computationally bounded or unbounded by their attack capabilities. They are also categorized as static or adaptive, depending on how they select targets for corruption.

1.3.2.3. Security definition

The dissertation applies a standard definition of security for multi-party computation protocols within the semi-honest model, utilizing the public communication channels of O. Goldreich. [2].

Key techniques in SMC include oblivious transfer, homomorphic encryption, and secret sharing.

1.3.3. Data obfuscation techniques

Data obfuscation techniques involve altering or generating data from the original dataset to train a model. These include additive and multiplicative perturbation, generative obfuscation, and data synthesis.

1.4. Privacy-Preserving Deep Learning: A Review

The literature examines three approaches to address these challenges. The first approach involves sharing local datasets directly in noisy or encrypted forms, followed by specialized learning algorithms [3–7]. This method, known as the ”data sharing approach,” utilizes techniques like homomorphic encryption (HE), SMC, secret sharing, or adding noise.

PATE [8] is another approach in privacy preserving deep learning. In this second approach, instead of sharing local training datasets, participants or ”teachers” share their knowledge of the predictive output to a ”student” server model. Then the ”student” server trains the student public model by using a public unlabeled dataset on the ensemble results from teacher models.

Distributed learning, particularly federated learning, is the predominant method for training deep learning models today. It addresses direct data leaks by exchanging intermediate training models instead of sharing local data directly. However, sharing model parameters directly can introduce vulnerabilities to indirect data leakage through attacks like model inversion or membership inference. As a result, studies have integrated techniques such as DP and SMC to enhance the security of sharing model parameter vectors.

DP methods often require a trade-off between model accuracy and privacy. Less noise enhances training model accuracy but raises vulnerability to attacks causing indirect data leakage. Consequently, there's promise in utilizing SMC in Federated Learning. However, SMC in Federated Learning faces two notable limitations.

- The first limitation concerns the need for participants to share the same key, rendering SMC vulnerable to collusion scenarios.
- The second limitation relates to efficiency in handling floating-point real number. Parameter vectors require conversion to large integers, significantly restricting computational capabilities of SMC protocols.

Thus, there's a pressing need to devise SMC protocols adept at handling collusion among multiple parties and maintaining accuracy with floating-point real number vectors in Federated Learning settings. This thesis seeks to address these limitations by proposing practical SMC protocols for preserving privacy during distributed training of deep learning models within Federated Learning frameworks. The main goal is to introduce *SMC protocols capable of effectively operating with floating-point real number vectors in distributed multi-party environments, even in the presence of collusion.*

1.5. Comparison of the PPDL Approach and Existing Limitations

The input sharing approach, aimed at enhancing security, often involves noise addition or cryptography. However, noise addition weakens security by making data susceptible to inference attacks and reduces model accuracy due to data distortion. While SMC improves security, it poses sig-

nificant challenges to model adaptation, increasing computational and communication complexity. It also relies on key sharing, limiting security to two-party computations, making it more suitable for prediction than training.

Output sharing impacts model accuracy due to errors from teacher models and requires public data and high-quality local models, which is impractical in distributed training environments with limited data.

Model sharing, split into split learning and federated learning, offers different trade-offs. Split learning shares parameters across specific layers but struggles with accuracy and participant limitations due to information leakage. Federated learning, however, is the most practical solution for distributed deep learning, balancing accuracy and execution cost while preventing direct data leakage. Still, it remains vulnerable to indirect leakage via exposed model parameters. To mitigate this, techniques like Differential Privacy (DP) and SMC are proposed, though DP sacrifices accuracy. Federated Learning combined with SMC emerges as a promising research direction, offering a balance between security and performance.

However, integrating Federated Learning with SMC faces key challenges:

- Participants must share cryptographic keys directly or via a trusted intermediary, which is vulnerable to collusion.
- Real number transformation into large integers increases computational load and slows down both calculation and data transmission.

1.6. Chapter Summary

This chapter has discussed the problem of ensuring privacy for deep learning, various approaches, and the pros and cons of each approach. From there, the dissertation identifies a focus on researching the problem of ensuring privacy for the training process of distributed deep learning networks, or more specifically, federated learning models. Through analysis, the dissertation also concludes that this training process essentially requires computing the sum of real-number vectors. Therefore, the dissertation will propose efficient protocols for computing the sum of real-number vectors to serve this purpose.

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

This chapter introduces three robust protocols for securely aggregating real-valued vectors, designed to withstand collusion. The detailed documentation of these protocols can be found in **Publications 3, 5, 6**, and **7**.

2.1. Cryptography preliminaries

This research is based on two crucial foundations in the field of cryptography, namely the discrete logarithm problem on elliptic curves and on finite fields.

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

2.2.1. Proposed protocol

The proposed protocol is summarized in Fig. 2.1.

2.2.2. Estimation error evaluation

Theorem 2.2.1. *The proposed protocol can approximate the sum of n vectors with the error bound of each j-th component calculated by the formula* $\Delta S^{(j)} = \sqrt{(\delta_1^{(j)})^2 + (\delta_2^{(j)})^2 + \ldots + (\delta_n^{(j)})^2} \leq d(n+1)$. *Here, d is the number of decimal digits for rounding.*

2.2.3. Privacy analysis

Theorem 2.2.2. *The protocol for secure n-clients sum presented in Fig protects each honest client' privacy against the server and up to* $(n-2)$ *corrupted clients.*

2.2.4. Performance Evaluation

2.2.4.1. Computational cost

The computational overhead for generating shared values and the time cost of performing secure aggregation (based on the Shank algorithm) are depicted in Figure 2.2. The results demonstrate that the protocol incurs low execution costs, making it well-suited for practical real-world application scenarios.

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: $x_i = \{y_i^{(j)}\}, y_i = \{y_i^{(j)}\}$.
- System parameters: the exponential factor (γ), $\mathbb{Z}_{\mathbf{p}}$ and generator g.

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

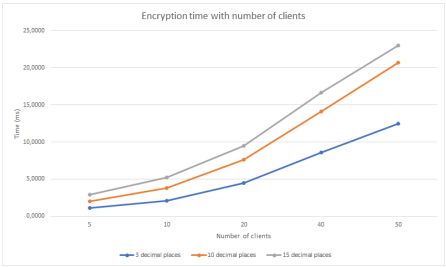
Phase 1: Initialization Phase

- 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.

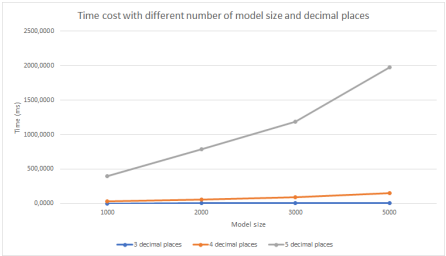
Phase 2: Main phase

- 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: $g^{S^{(j)}} = V^{(j)}$ 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



(a) The average compute share values time at decimal precision levels of 2, 3, and 5



(b) Computational time for the secure aggregation of different model size at decimal precision levels

Figure 2.2: Computation Cost at compute share values and secure aggregation phase with SVS1 Protolcol

2.2.4.2. Communication cost

The bandwidth consumption at the Client and Server for each round of the protocol is outlined in Table 2.1. The results indicate that the protocol demands four times the bandwidth compared to a model without privacy protection. However, this trade-off is likely acceptable in practical applications, given the enhanced privacy guarantees.

	Client i	**Server**
Round 1	$2 \times$ model size $\times$ key size	$2 \times$ model size $\times$ key size $\times\, n$
Round 2	model size $\times$ key size	model size $\times$ real number size $\times\, n$

Table 2.1: Communication Cost per Round

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

2.3.1. Proposed Protocol

Using an elliptic curve analog of the ElGamal system, the proposed scheme allows multi parties jointly compute the sum of their private messages without revealing the actual values. The proposed protocol is summarized in Fig. 2.3.

2.3.2. Proof of correctness

Theorem 2.3.1. *The proposed protocol in the Figure 2.3 can calculate the sum of n vectors.*

We prove that $\sum_{i=1}^{n} T_i$ is equal to $\sum_{i=1}^{n} W_i$, which implies that $T = \sum_{i=1}^{n} W_i$.

2.3.3. Privacy analysis

Theorem 2.3.2. *The protocol for secure n-clients sum presented in Figure 2.3 protects each honest client' privacy against the server and up to* $(n-2)$ *corrupted clients (and colluding with the server) in semi-honest model.*

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

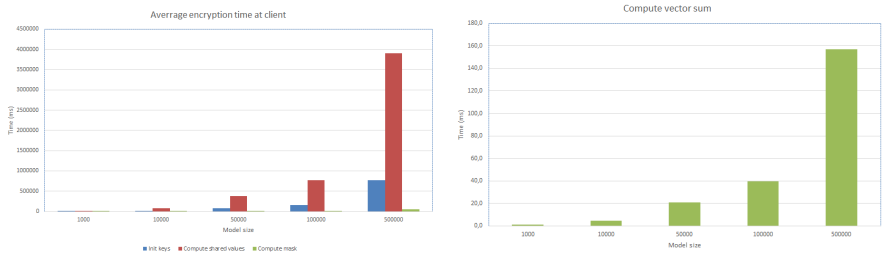
2.3.4. Performance evaluation

2.3.4.1. Computational cost

The computational overhead for generating shared values and the time cost of performing secure aggregation (based on the Shank algorithm) are depicted in Figure 2.4. The results demonstrate that the protocol incurs low execution costs, making it well-suited for practical real-world application scenarios.

2.3.4.2. Communication cost

The bandwidth costs for the Client and Server at each round of the protocol are presented in Table 2.2. The results indicate that this protocol demands significantly more bandwidth compared to the first proposed protocol. However, for scenarios requiring high precision, this protocol offers



(a) The average compute share values time

(b) Computational time for the secure aggregation

Figure 2.4: Computation Cost at compute share values and secure aggregation phase with SVS2 Protolcol

lower computational costs. Therefore, it remains a viable option for practical applications.

	Client i	**Server**
Round 1	$4 \times$ model size $\times$ key size	$4 \times$ model size $\times$ key size $\times\, n$
Round 2	$4 \times$ model size $\times$ real number size $+ 2 \times$ model size $\times$ key size	$2 \times$ model size $\times$ real number size $\times\, n$
Round 3	model size $\times$ real number size	model size $\times$ real number size $\times\, n$

Table 2.2: Bandwidth costs per round for Client and Server

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

2.4.1. Proposed protocol

The proposed protocol is summarized in Fig. 2.5.

2.4.2. Proof of correctness

Theorem 2.4.1. *The proposed protocol in the Figure 2.5 can calculate the sum of n vectors.*

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

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 2.5: SVS Protocol based on mask matrix and ElGamal cryptosystem

2.4.3. Privacy analysis

This section of the dissertation establishes that (i) each user U_i, with parameters P_i, r_i, s_i, is successfully authenticated, (ii) the protocol is provably secure against potential attacks within the random oracle model, and (iii) the protocol demonstrates resilience to collusion involving up to $n-2$ participants, including the aggregation server.

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

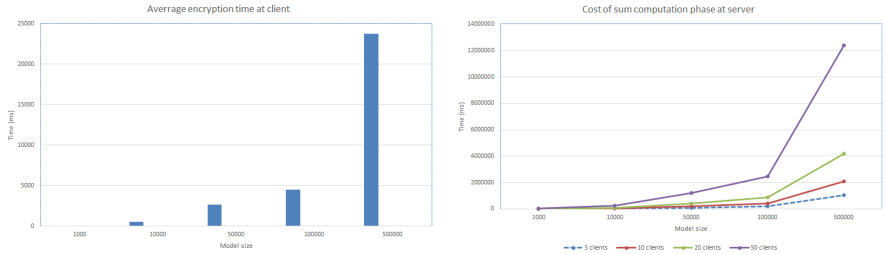
Theorem 2.4.3. *The multi-party sum protocol using El-Gamal protects the confidential data of any participating client even if there are n-2 colluding members (and colluding with the server).*

2.4.4. Performance evaluation

2.4.4.1. Computational cost

Figure 2.6 illustrates the computational overhead for generating shared values and the time required for secure aggregation using the Shank algorithm. The results show that the protocol maintains low execution costs,

making it an efficient and practical choice for real-world applications.



(a) The average compute share values time

(b) Computational time for the secure aggregation

Figure 2.6: Computation Cost at compute share values and secure aggregation phase with SVS3 Protolcol

2.4.4.2. Communication cost

The bandwidth requirements for the Client and Server in each round of the protocol are detailed in Table 2.3. The results show that the protocol requires four times the bandwidth of a model without privacy safeguards. However, this increase in bandwidth is a reasonable trade-off in real-world applications, as it ensures stronger privacy protection.

	Client i	**Server**
Round 1	$2 \times$ model size $\times$ key size	$2 \times$ model size $\times$ key size $\times\, n$
Round 2	model size $\times\, (2 \times$ key size $+$ real number size$)$	model size $\times$ real number size $\times\, n$

Table 2.3: Communication Cost per Round

2.5. Chapter Summary

This chapter has analyzed and proposed three new secure real-number summation protocols. The proposed protocols have been proven secure in the semi-honest model. The evaluations have also demonstrated their effectiveness. Therefore, the proposed protocols are capable of being applied to real-world problems that require secure computation of sum values or frequencies.

CHAPTER 3. DEVELOPING FEDERATED LEARNING SCHEME BASED ON PROPOSED SECURE MULTIPARTY SUM PROTOCOLS

3.1. Secure federated learning framework with semi-trusted server

3.1.1. Proposed framework

Here is a summary of the centralized framework embedding the secure sum protocols:

Framework 1: 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, 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 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 framework's operation is depicted in Framework 1, where **Phase (1) - Compute Public Share Values** and **Phase (2) - Secure Sum Computation** represent the execution stages of the SMC protocols in Chapter 2.

3.1.2. Experimental setup

Experiments were conducted on three datasets—CSIC2010, MNIST, and SMS-Spam—to examine the impact of various factors on the global model's performance, using CLCNN, CNN, and LSTM as the respective network architectures.

3.1.3. Experimental results and evaluation

3.1.3.1. The overall model performance

Centralized federated learning framework with secure multi-party sum protocol using masking matrix

- *Varying the number of the clients.* The first experiment involves evaluating the performance of the global model under varying numbers of clients. The ElGamal multiparty secure sum protocol was employed to train the framework over 50 communication rounds. Figure 3.1 presents the study outcomes with three datasets: CSIC2010, MNIST, and SMS-Spam.

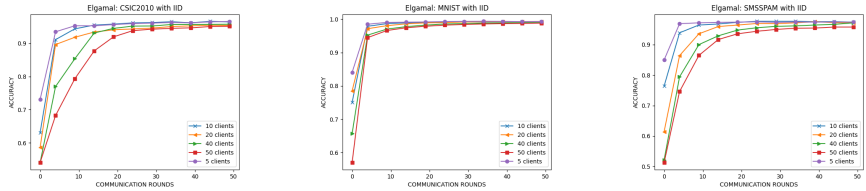


Figure 3.1: The results on accuracy with different number of clients.

- *Varying the dropout rate at each communication rounds*. Figure 3.2 shows the impact of different dropout rates on the model's performance. Systematic evaluations were performed at dropout rates of 0%, 10%, 20%, 30%, 40%, and 50%, respectively.

Upon careful examination, it is evident that the outcomes obtained using the secure sum protocol employing ElGamal cryptography closely align

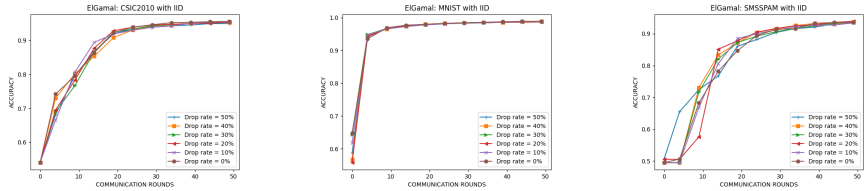


Figure 3.2: The results on accuracy with different dropout rates.

with the results from experimenting with the secure sum protocol using ECC cryptography. This alignment is attributed to preserving model parameters in their original float number format, ensuring the integrity of the total number of desired models.

Centralized federated learning framework with secure multi-party sum protocol using quantization technique

- *Impact of the number of decimal places*

  As illustrated in Figure 3.3, it becomes evident that rounding the model parameters to a mere 3 digits results in a substantial reduction in accuracy. Conversely, employing higher encoding precision settings (i.e., 5 and 10 decimal places) yields only negligible variations in performance. It is worth noting that lower encoding precision not only compromises model accuracy but also adversely affects the convergence rate, with the model requiring additional time to converge.

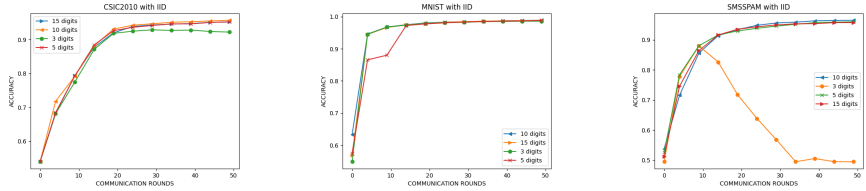


Figure 3.3: The result on accuracy for different precision levels.

When the decimal place is fixed 3, Figure 3.4 shows the accuracy of the framework with the variation of the number of clients.

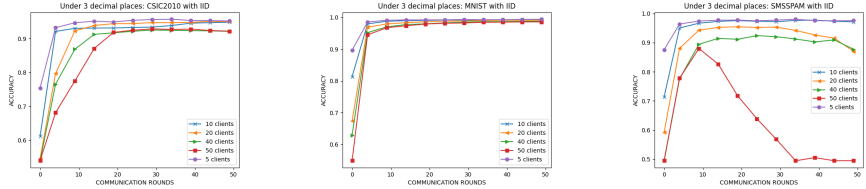


Figure 3.4: The accuracy value with 3 decimal places when varying the number of clients.

3.1.3.2. The overall performance when the distribution of data across clients is non-IID

- *Varying the number of the clients.* Figure 3.5 show that increasing the number of clients does not necessarily lead to better performance. As the number of clients increases, the diversity of the data may increase. However, the challenges associated with coordinating the clients and aggregating their updates also increase. This is visible in the 50 clients case where the model seems to struggle to find a good convergence pattern.

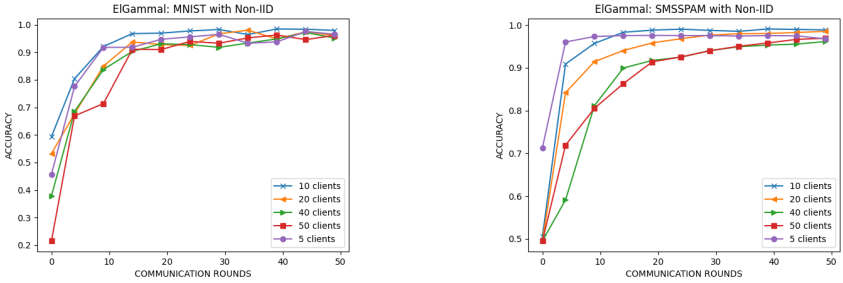


Figure 3.5: The results on accuracy and loss with different number of clients with non IID.

- *Varying the dropout rate at each communication rounds.* Figure 3.6 show the accuracy of the global model with different dropout rate with non IID.

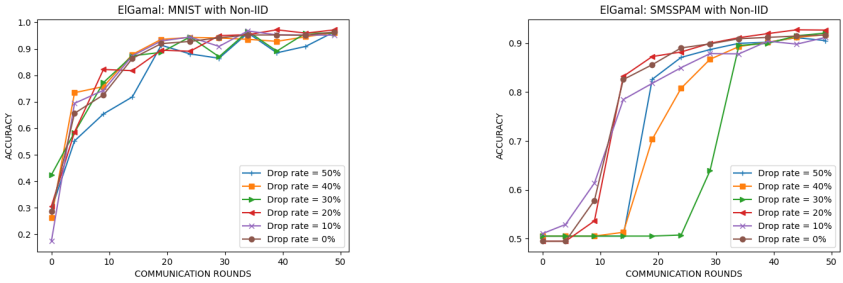


Figure 3.6: The results on accuracy with different dropout rates.

3.1.3.3. Comparing the accuracy of the framework with other strategies.

The obtained results are shown in Table. 3.1.

Table 3.1: Compare performance with different strategies

Method	**Data**					
	IID			**Non-IID**		
	MNIST	**CSIC 2010**	**SMS Spam**	**MNIST**	**CSIC 2010**	**SMS Spam**
Standalone	.9658	.9448	.9203	.314	.6103	.8821
Centralized	.9943	.9676	.9806	.9943	.9676	.9806
Fed-Avg (small noise)	.9741	.9469	.9147	.9454	.9442	.8173
Fed-Avg (large noise)	.9659	.8745	.6397	.8954	.7214	.5634
Selective learning	.9629	.9575	.9619	.9724	.8534	.9662
FedAvg + SMC 1	.9942	.9685	.9728	.9882	.9433	.9762
FedAvg + SMC 2	.9913	.9654	.9692	.9869	.9621	.9793
FedAvg + SMC 3	.9913	.9654	.9692	.9869	.9621	.9793

3.2. Secure federated learning framework in decentralized network settings

3.2.1. Proposed framework

This section implements three security protocols within the decentralized, federated learning framework. Here is a summary of the framework:

Framework 2: 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.

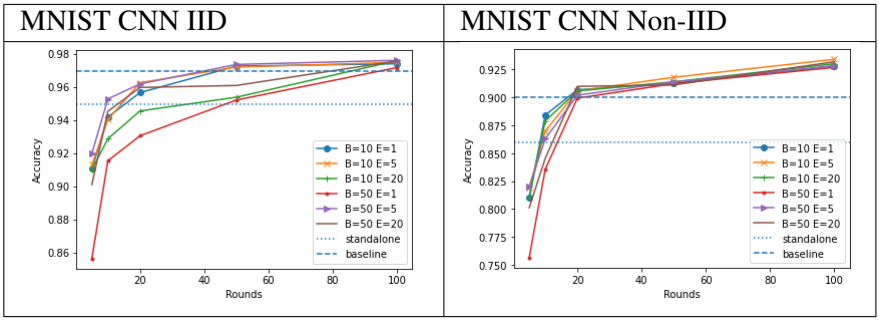
3.2.2. Experimental setup

Experiments were conducted using the MNIST and UCI SMS Spam to assess the proposed framework. This section presents the efficiency of the proposed framework in handling IID, Non-IID, and imbalanced data.

3.2.3. Experimental results and evaluation

Overall model performance. Table 3.2 shows the performance of the Secure Decentralized Training Framework (SDTF). After 100 communication rounds, the framework achieved 97.6% accuracy on the IID dataset and 93% on the Non-IID dataset, demonstrating its strong performance and high accuracy despite variations in local parameters.

Table 3.2: The results on accuracy for different local training parameters on MNIST CNN (IID and Non-IID)



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 cooperation results are presented in Table 3.3. 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.

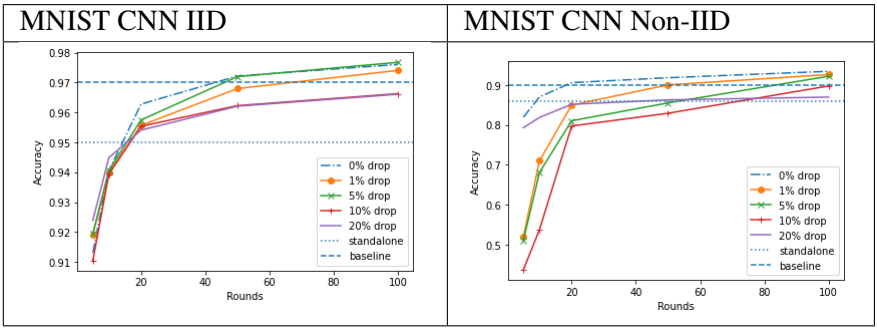
Table 3.3: Model accuracy comparison: balanced dataset

	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

Impact of Dropped Nodes Ratio on Performance. This section evaluates the proposed framework's performance under varying node dropout ratios. With a fixed batch size of 5 and 10 local epochs, the evaluation occurs

at the 5th, 10th, 20th, 50th, and 100th rounds. Dropout ratios of 1%, 5%, 10%, and 20% are tested, corresponding to 99%, 95%, 90%, and 80% node participation per round. The results are shown in Table 3.4.

Table 3.4: Results on the framework's performance for the dropped nodes on the MNIST dataset.



Imbalanced datasets. 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 3.5.

Table 3.5: Model accuracy comparison: imbalanced dataset

	Selective 10%	Selective 50%	Downpour SGD	FedAVG	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

3.3. Chapter Summary

This chapter introduced two distributed deep learning training protocols based on the secure real-number vector summation methods from the previous chapter. Evaluation on the MNIST, SMS Spam, and CSIC 2010 datasets showed that the resulting models achieved high accuracy, outperforming other methods, demonstrating the protocols' practical applicability. These findings are detailed in **Publications [2,3,4,5,6,7]**.

CHAPTER 4. CONCLUSION AND FUTURE WORK

Deep learning has emerged as a powerful tool in various machine learning domains, including image classification, speech recognition, natural language processing (NLP), and bioinformatics. However, leveraging deep learning effectively relies heavily on access to large amounts of data for training. Federated learning, pioneered by Google Brain, offers a solution by allowing training data to remain on local devices while a shared model is learned through aggregated updates.

However, sharing model parameters in federated learning can inadvertently compromise user privacy. Current research focuses on addressing this issue through methodologies such as differential privacy or cryptographic-based approaches like secure multiparty computation (SMC). While cryptographic approaches offer promising solutions for enhancing both security and efficiency, challenges remain, particularly in converting floating-point real numbers into integers, which is computationally demanding and can lead to accuracy loss.

To address these challenges, this thesis proposes three secure multiparty sum protocols tailored for floating-point real number vectors in semi-honest collision scenarios. These protocols ensure differential privacy and computational security while minimizing communication and processing costs. Despite their high efficiency, they lack data authentication and are vulnerable to membership spoofing attacks. Therefore, a novel protocol combining random noise masking, the ElGamal cryptosystem, hashing methods, and digital signatures is introduced to tackle these issues.

Empirical evaluations on various datasets and deep neural network architectures demonstrate the proposed methodology's effectiveness, achieving high accuracy levels even in distributed networks with non-IID and imbalanced data distributions. Furthermore, the proposed protocols are designed to withstand collusion among $n-2$ parties, ensuring privacy in privacy-preserving deep learning.

Looking ahead, it is essential for the research community to develop new SMC protocols tailored to evolving distributed computing scenarios and future cryptographic advancements, such as post-quantum cryptography. Additionally, integrating SMC into input sharing approaches and ensemble learning can further enhance efficiency and address specific training model cases under certain conditions.

LIST OF THE PUBLICATIONS RELATED TO THE DISSERTATION

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).

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, 2024. (ISI indexed, Scopus).