École Polytechnique Fédérale de Lausanne

Nagra Kudelski Group

# Privacy-Preserving Cross-Device Federated Learning Based on Multiparty Homomorphic Encryption with Hierarchical Access Structure

by Haowen Liu

## Master Thesis

Approved by the Examining Committee:

Prof. Boi Faltings
Thesis Advisor

Hedi Fendri
External Expert

Luca Gradassi
Thesis Supervisor

EPFL IC IINFCOM LIA
INR 230 (Bâtiment INR)
Station 14
1015 Lausanne

August 16, 2024

# Acknowledgments

It would not have been feasible to pen this thesis without the support of the good people around me, to only some of whom it is possible to give a particular mention here.

Firstly, I would like to express my sincere gratitude to the **Kudelski Group** and **Swiss Federal Institute of Technology in Lausanne (EPFL)** for letting me perform my master's thesis.

To my supervisors, Mr. **Luca Gradassi** senior security expert at Kudelski group, Prof. **Boi Faltings** full professor at EPFL, Mr. **Hedi Fendri** senior data scientist and Mr. **Aymen Bahroun** data scientist at Kudelski Group, I am extremely grateful for your assistance and suggestions throughout my project.

Special thanks to my colleagues at Kudelski group **Karine Villegas** for providing me technical support and helping me to learn many things, **Evelyne Taureg** for helping me onboard and get familiar with the working environment.

Finally, I would like to acknowledge with gratitude the support and help of my parents, **Qinjian** & **Jinlan**, and my friends **Jiaxun**, **Youssef**, **Xiao**, **Yuan**, **Xiangcheng**, **Duo**, **Jacomo**, and **Xiaoyu**. During the course of this graduation project, I went through a very difficult time. It was their companionship that guided me out of the darkness and into the light. They all kept me going, and this thesis would not have been possible without them, thank you very much.

*Lausanne, August 16, 2024* Haowen Liu

# Abstract

In the domain of machine learning, Federated Learning (FL) emerged as a pivotal strategy to mitigate data leakage by processing updates from clients rather than exposing raw data. Nonetheless, vulnerabilities such as gradient inversion attacks have underscored the limitations of conventional FL approaches, revealing the potential for data leakage from updates. This challenge has prompted the exploration of privacy-preserving mechanisms, notably through Differential Privacy (DP), Secure Multiparty Computation (SMC), Secure Aggregation (SA), and Homomorphic Encryption (HE). Our project zeroes in on enhancing privacy without compromising model accuracy by leveraging HE, supported by SMC, within the framework of FL.

We identify the adaptation of HE to FL as critical, especially in the context of Internet of Things (IoT) scenarios characterized by **large-scale** and easy-to-access client devices, **heterogeneous** delays, and **unreliable** networks. Traditional HE schemes, designed for cross-silo FL, falter in addressing these IoT-specific challenges. To bridge this gap, we propose a pioneering Hierarchical Multiparty Homomorphic Encryption (H-MHE) protocol tailored for Hierarchical FL (HFL). Our approach innovates by extending Threshold Multiparty HE (TMHE) to a hierarchical access structure, aiming to establish a precedent for effective and practical privacy-preserving FL in IoT environments.

We encompassed a comprehensive investigation, implementation, and testing of the H-MHE protocol, marking a significant stride toward securing FL in IoT applications. By addressing the nuanced requirements of IoT scenarios, our work aspires to fortify the privacy landscape of FL, ensuring robust data protection while maintaining model performance.

In our experiments, we simulated a 100-device network and tested H-MHE with 4 clusters in the network, with MNIST dataset classification as the FL task. H-MHE achieved a **12.25**× speedup during cryptosystem setup phase and 1.14× speedup during training phase compared to the state-of-the-art TMHE framework, while maintaining asymptotic resilience against network unreliability. Notably, the trained model maintains its accuracy compared to models trained by conventional raw FL framework without protection. The evaluation demonstrates that the scheme outperforms other innovations in resilience to unreliable networks, communication cost, and computational overhead, while preserving model accuracy in cross-device FL scenarios.

# Contents

**Acknowledgments** **2**

**Abstract** **3**

**List of Figures** **6**

**List of Tables** **7**

**1 Introduction** **10**
1.1 Problem Statement . . . 10
1.2 Existing Solutions . . . 11
1.3 Contribution . . . 12
1.4 Thesis Organization . . . 13

**2 Background** **14**
2.1 Machine Learning Overview . . . 14
2.1.1 Machine Learning . . . 15
2.1.2 Federated Learning . . . 15
2.1.3 Privacy Concerns . . . 16
2.2 Internet of Things Overview . . . 17
2.3 Homomorphic Encryption Preliminaries . . . 18
2.3.1 Definition . . . 18
2.3.2 Partially Homomorphic Encryption . . . 19
2.3.3 Fully Homomorphic Encryption . . . 20
2.3.4 Security Level . . . 21

**3 Design** **22**
3.1 Threat Model . . . 22
3.2 Multiparty Homomorphic Encryption . . . 23
3.2.1 Plain Multiparty Homomorphic Encryption . . . 23
3.2.2 Fully-Multiparty Homomorphic Encryption . . . 24
3.2.3 Threshold Multiparty Homomorphic Encryption . . . 26
3.3 FL with HE . . . 27

3.4 Hierarchical Multiparty Homomorphic Encryption Protocol with RLWE-Based Scheme 30
3.5 Integrating HFL with H-MHE . . . . . 32

**4 Implementation and Evaluation** **34**
4.1 Utility Basics . . . . . 35
4.1.1 `Lattigo v5.0.2`: Lattice-Based Multiparty Homomorphic Encryption Library in Go . . . . . 35
4.1.2 `go-deep`: Feed Forward/Back-Propagation Neural Network Implementation . 35
4.2 A Privacy-Preserving Federated Learning Demo with Hierarchical Multiparty Homomorphic Encryption Protocol using RLWE-Based Scheme . . . . . 36
4.2.1 HFL Framework . . . . . 36
4.2.2 Multiparty Homomorphic Encryption with Hierarchical and Hybrid Access Structure . . . . . 38
4.2.3 Integration of Hierarchical Multiparty Homomorphic Encryption Protocol and Federated Learning . . . . . 39
4.2.4 Security Analysis . . . . . 45
4.2.5 Performance Analysis of Device Layer . . . . . 45
4.2.6 Performance Analysis of Gateway Layer . . . . . 46
4.2.7 Performance Analysis of Server Layer . . . . . 47
4.3 Experiments . . . . . 47
4.3.1 Heterogeneous Large-scale Network Simulation . . . . . 47
4.3.2 Federated Learning Setting . . . . . 48
4.3.3 Experimental Results . . . . . 48

**5 Conclusion and Future Work** **51**

**Bibliography** **53**

# List of Figures

1.1 A Gradient Inversion attack example targeting a neural network. The left is the private data point, the middle is the reconstructed sample from gradients by Gradient Inversion attack [73], and the right is the reconstruction error. . . . . 11

3.1 An example of PMHE on Federated Learning application. . . . . 24

3.2 An example of FMHE on Federated Learning application. . . . . 25

3.3 Using share-resharing technique to transfer FMHE to TMHE with threshold $T$ out of $N$. 26

3.4 Design of H-MHE on Federated Learning application. . . . . 33

4.1 System architecture for a three-tier IoT networks running FL applications with heterogeneous delays and unreliable networks [70]. . . . . 37

4.2 Hierarchical Homomorphic Encryption Framework for Hierarchical Federated Learning on IoT Networks. . . . . 38

# List of Tables

4.1 Heterogeneity setting of the network simulation. We divide the network to four categories, each contains 25 devices. For devices in each category, their dropout rate are sampled from the normal distribution $\mathbb{N}(\mu, \sigma^2)$. The last column is the thresholds of H-HME clusters according to the cluster network setting. . . . . 48

4.2 Model Architectures, hyperparameters, and distributed settings for MNIST experiments. L(#*outputs*) for a fully-connected linear layer with #*outputs* dimensions of output, R for ReLU activation. . . . . 49

4.3 Experimental results about best accuracy, time overhead during cryptosystem setup and training, and the number of rounds failing to decrypt. Each experiment runs 5 times and the average is shown in the table. . . . . 49

# Nomenclature

| | |
|---|---|
| $\mathbb{P}$ | Party set |
| $\mathbb{P}_{online}$ | Online party set |
| $n$ | RLWE dimension |
| $q$ | ciphertext modulus |
| FL | Federated Learning |
| HE | Homomorphic Encryption |
| ML | Machine Learning |
| CKKS | Cheon-Kim-Kim-Song Scheme |
| CRS | Common Random String |
| CRT | Chinese Remainder Theorem |
| DP | Differential Privacy |
| FHE | Fully Homomorphic Encryption |
| FMHE | Fully-Multiparty Homomorphic Encryption |
| H-MHE | Hierarchical Multiparty Homomorphic Encryption Framework |
| HFL | Hierarchical Federated Learning |
| MHE | Multiparty Homomorphic Encryption |
| NTT | Number-Theoretic Transform |
| PHE | Partially Homomorphic Encryption |
| PKI | Public Key Infrastructure |
| PMHE | Plain Homomorphic Encryption |

RLWE Ring Learning With Errors

SA Secure Aggregation

SMC Secure Multiparty Computation

SR Share-Resharing

TMHE Threshold Multiparty Homomorphic Encryption

# Chapter 1

# Introduction

In recent years, the proliferation of data-driven technologies has underscored the critical importance of privacy-preserving mechanisms in machine learning (ML) [28, 67]. Among the various strategies devised to protect user data, Distributed ML has emerged as a forefront approach [1, 5, 15, 42, 58]. Distributed ML breaks a complex ML task into sub-tasks that are performed in a collaborative fashion [3]. In the standard server-based architecture, which we also call Federated Learning (FL), several machines collaboratively train a global model on their datasets, with the help of a coordinator (the server) [3].

In FL, the server maintains a model which is updated iteratively by aggregating model updates from clients. The model updates can be 1) gradients of the loss function associated with the model, computed by the different workers upon sampling random points from their local datasets (FedSGD) [42]; 2) updated model parameters after applying the aforementioned gradients on the transient models for one or several training iterations (FedAVG) [42]. Distributed ML is particularly useful in cases where the data held by the workers is too sensitive to be shared, e.g., medical data collected by several hospitals [58].

## 1.1 Problem Statement

Although FL inherently ensures the privacy of the workers' data to an extent, by not sharing it explicitly, information leakage can still be significant. When the ML model maintained at the server is publicly released, it may be exposed to membership inference [60] or model inversion attacks [22, 27, 43] by external entities. Furthermore, upon observing the model updates and transient models during the learning procedure, curious machines (clients or the server itself) can infer sensitive information about the datasets held locally by the machines, or even reconstruct data points in certain scenarios [54, 65, 72, 74], shown as Figure 1.1.

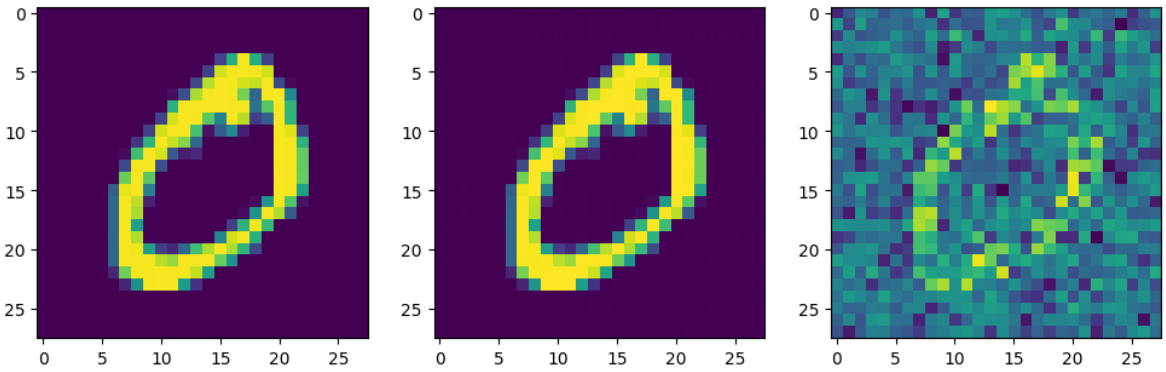

Figure 1.1: A Gradient Inversion attack example targeting a neural network. The left is the private data point, the middle is the reconstructed sample from gradients by Gradient Inversion attack [73], and the right is the reconstruction error.

## 1.2 Existing Solutions

To mitigate the privacy leakage due to exposure of model updates, lots of defenses were introduced to FL. These defenses can be categorized to two directions: noise-based defenses, including Differential Privacy (DP) [14, 29, 50] and Masking [66], and crypto-based defenses, including Secure Aggregation (SA) [7, 21, 32, 61], Secure Multi-party Computation (SMC) [44, 45], and Homomorphic Encryption (HE) [11, 30, 41, 53, 71].

DP often suffers from accuracy decline due to noise added [53, 71], yet it can be a good defense against general privacy leakage vulnerabilities in ML. Masking usually requires external trusted authorities [66]. SA, SMC, and HE are designed to defend against attacks targeting computation process, so also can be weak to attacks targeting the final model, especially in Cross-device FL scenarios, since the adversary can simply register a (cheap) device to acquire the transient model in a legitimate way. More specifically, SA, SMC, and HE all enable secret computation on model updates, while SA will expose the results to the server, but SMC and HE may not. And SA can utilize non-crypto scheme, while SMC and HE leverage cryptosystems to protect model updates. So both SMC and HE provide strong privacy protection during computation based on the hard problem their cryptosystems use.

Note that current defenses often use a combination of aforementioned techniques to provide comprehensive protection as well as good efficiency, and so as our proposal. However, none of the existing solutions are practical on a zero-trust (in this thesis, zero-trust entities do not include Public Key Infrastructure (PKI) for secure communication and authentication), large-scale, heterogeneous, and unreliable network, e.g., IoT networks.

## 1.3 Contribution

In this thesis, we focus on leveraging HE as a defense mechanism, supported by SMC, within the context of cross-device Federated Learning (FL). We introduce a novel Hierarchical Multiparty Homomorphic Encryption Framework (H-MHE) tailored for Hierarchical Federated Learning (HFL) [38, 39, 70], building upon existing multiparty homomorphic encryption protocols designed for distributed systems. Collectively referred to as Multiparty Homomorphic Encryption (MHE) protocols, these methodologies enable a system to operate as an abstract encryption and decryption engine using a virtual public-private key pair. Examples of such protocols include Plain Multiparty Homomorphic Encryption (PMHE) [26], Fully-Multiparty Homomorphic Encryption (FMHE) [41, 49], and Threshold Multiparty Homomorphic Encryption (TMHE) [18, 49].

The H-MHE framework extends these abstracted encryption engines into a hierarchical architecture, making it inherently scalable across multiple layers. For instance, a three-tier H-MHE system (with one level of abstraction) can be seamlessly adapted into a three-tier HFL system for privacy-preserving machine learning applications.

This scheme offers two primary advantages: 1) At the Device Layer, we employ TMHE and PMHE to tolerate device dropout and heterogeneous delays in IoT networks while maintaining a threshold-based privacy guarantee; 2) At the Gateway Layer, we adapt FMHE, leveraging the fact that gateways are typically more stable than Internet of Things (IoT) devices. This allows us to use the all-party-participation requirement of FMHE to provide a higher level of privacy, while mitigating the drawback that FMHE is less tolerant of dropout and delay.

Furthermore, we propose a practical implementation of the H-MHE framework based on the Ring-Learning-With-Errors (RLWE) encryption scheme. This implementation is designed as a three-tier H-MHE framework where client devices are organized into clusters during the setup phase. Each cluster is assigned a gateway, which functions as an auxiliary sub-server for the cluster and represents it in communications with other clusters. Within each cluster, an MHE protocol is executed, while an abstract FMHE protocol is run among the gateways of all clusters, potentially with the assistance of a central server. All MHE protocols operate under the same public parameter settings and a shared common random string.

We conduct comprehensive experiments on a simulated large-scale, unreliable, and heterogeneous network, to evaluate the correctness and performance of the FL with H-MHE prototype system in comparison with current state-of-the-art multiparty homomorphic encryption schemes, and further test its tolerance against dropouts.

## 1.4 Thesis Organization

The remainder of this thesis is organized as follows:

- **Chapter 2 - Background:** This chapter lays the foundation for the thesis by introducing the key concepts and technologies relevant to this work. It is divided into three main sections: an overview of machine learning (ML) including its types and privacy challenges, an exploration of the Internet of Things (IoT) and its security concerns, and a detailed discussion on homomorphic encryption and its applications in privacy-preserving computations.
- **Chapter 3 - Design:** In this chapter, we delve into the design decisions and methodologies that underpin the proposed solution. It covers the development of multiparty homomorphic encryption schemes tailored for federated learning, including a detailed comparison of different approaches and the introduction of the Hierarchical Multiparty Homomorphic Encryption (H-MHE) protocol.
- **Chapter 4 - Implementation and Evaluation:** This chapter provides an in-depth account of the implementation of the H-MHE framework and its integration with federated learning. It includes a comprehensive evaluation of the framework's performance through various experiments, assessing factors such as model accuracy, communication overhead, and resilience in large-scale IoT environments.
- **Chapter 5 - Conclusion and Future Work:** The final chapter summarizes the key findings of the thesis, highlighting the contributions of the proposed H-MHE protocol to privacy-preserving federated learning in IoT contexts. It also discusses potential avenues for future research, including the integration of differential privacy and the optimization of homomorphic encryption for real-world IoT applications.

# Chapter 2

# Background

In this chapter, we establish the key concepts and technologies foundational to this thesis, structured into three sections addressing crucial areas. The first section introduces machine learning (ML), covering its core categories: supervised, unsupervised, and reinforcement learning, alongside advanced topics like deep learning and federated learning. It also addresses privacy challenges, particularly in federated learning, laying the groundwork for later discussions on privacy-preserving techniques. The second section examines the Internet of Things (IoT), a rapidly expanding network of interconnected devices. We explore its key features, applications, and the associated security and privacy concerns, as well as the role of ML in managing the vast data generated by IoT devices. The final section focuses on homomorphic encryption, a technique enabling computations on encrypted data. We discuss its various forms, including partially and fully homomorphic encryption, and address security levels, especially in the context of emerging challenges like quantum computing.

## 2.1 Machine Learning Overview

In this section, we provide a overview of machine learning. The discussion begins with a broad introduction to the concept of machine learning, including its primary categories and applications. We then explore federated learning, an emerging technique that addresses privacy concerns by decentralizing the training process. Finally, we examine the privacy challenges associated with machine learning and federated learning, highlighting the need for robust privacy-preserving techniques. Each of these sections builds on the previous, providing a layered understanding of the current landscape of machine learning.

### 2.1.1 Machine Learning

Machine learning (ML) is a fundamental subset of artificial intelligence (AI) that empowers systems to autonomously learn from data and enhance their performance over time without explicit programming. ML involves the development of algorithms that can process input data, perform statistical analyses, and make predictions, all while continuously refining these predictions as new data is introduced. The applications of ML are extensive and diverse, spanning areas such as email filtering, speech recognition, and complex decision-making systems used in autonomous vehicles and financial modeling [31].

ML models are generally classified into three main categories: supervised learning, unsupervised learning, and reinforcement learning [37]. Supervised learning involves training models to predict outputs based on labeled input data, making it highly effective for tasks like classification and regression [35]. In contrast, unsupervised learning focuses on uncovering hidden patterns or structures within unlabeled data, which is particularly useful in clustering and association tasks [56]. Reinforcement learning, the third category, is based on the concept of an agent learning by performing actions and receiving feedback on the outcomes, which has broad implications for sequential decision-making problems in both industry and academia [62]. Each of these learning paradigms addresses distinct types of problems and continues to drive innovation and research in machine learning.

Deep learning, a specialized branch of machine learning, organizes algorithms into multiple layers to create what is known as an neural network, thereby enabling systems to learn and make decisions autonomously. This hierarchical structure allows deep learning models to effectively manage large datasets and identify intricate patterns that would be challenging for a human programmer to explicitly define [35]. As a result, deep learning has become the technique of choice for a variety of tasks that require advanced pattern recognition and feature extraction, including speech recognition, image recognition, and natural language processing.

### 2.1.2 Federated Learning

Federated learning is an innovative machine learning technique that enables multiple decentralized devices or servers to collaboratively train a shared model while keeping the training data on local devices. This method effectively decouples the ability to perform machine learning from the need to store data centrally, which is particularly advantageous in scenarios where data privacy, access rights, or bandwidth constraints are critical concerns [42]. Federated learning is particularly useful in situations where data cannot be transferred to a central server due to privacy regulations or security reasons, making it a powerful tool for distributed ML [33].

Federated learning can be broadly categorized into two types:

**Cross-Silo Federated Learning:** This variant is typically used by organizations such as hospitals or financial institutions, where large volumes of data exist but cannot be shared due to regulatory or competitive reasons. Cross-silo federated learning enables these organizations to jointly train models without sharing the underlying data, thereby preserving privacy and enabling the use of collective insights without compromising sensitive information [68]. This approach has seen significant applications in the healthcare sector, where patient data must remain confidential while benefiting from shared knowledge across institutions [55].

**Cross-Device Federated Learning:** This variant is more common in scenarios involving a vast number of devices, such as smartphones or IoT devices. Cross-device federated learning facilitates the training of models across millions of devices, each potentially contributing small datasets. However, this approach faces several challenges, including high communication overhead, devices going offline, and the need to scale to accommodate a large number of participants [6]. Primary concerns in this domain include scalability, communication efficiency, device heterogeneity, and privacy preservation. Recent research has focused on mitigating these challenges by optimizing communication protocols, developing lightweight model architectures, and implementing advanced privacy-preserving techniques, making cross-device federated learning more feasible and effective for real-world applications [70].

### 2.1.3 Privacy Concerns

Machine learning models inherently rely on large volumes of data for effective training, which raises significant privacy concerns, especially when dealing with sensitive personal information. Privacy risks in ML can manifest in several ways:

- Data Exposure: During the training process, sensitive data must be accessible to the algorithms, which can potentially expose personal information to data scientists and others with access to the system [59].
- Inference Attacks: Even without direct access to the training data, adversaries can perform inference attacks, such as model inversion or membership inference, to extract sensitive details about the training data or determine whether specific data was used in the training process [9, 69].
- Model Theft: By gaining access to the trained model, attackers can reconstruct, steal, or misuse the underlying data or the intellectual property embedded within the model [63].

The trade-off between model performance and data privacy often leads organizations and researchers to prioritize access to rich datasets, sometimes at the expense of user privacy. This

tension has spurred the development of privacy-preserving techniques, such as data anonymization, differential privacy, and secure multiparty computation, each with its strengths and limitations in safeguarding user data [64].

Federated learning, by design, addresses some of the privacy issues inherent in traditional ML by ensuring that data remains on the local device. However, this approach does not entirely eliminate privacy concerns:

- Information Leakage during Training: Even though data does not leave its original location, information can still leak through the shared model updates. For example, gradients shared during training can potentially reveal underlying data through differential attacks such as gradient inversion attacks [74].
- Collusion Threats: Participants in the FL process might collude to reconstruct sensitive data from other participants by analyzing shared gradients or updates.

Cross-device FL scenarios present additional challenges due to the scale and heterogeneity of the devices involved. Devices like smartphones, which vary in computational power and security, might be more vulnerable to breaches, increasing the risk of data exposure during computation or communication.

Addressing these concerns requires robust cryptographic techniques, strict data handling policies, and potential innovations in both ML and cryptography. Homomorphic encryption, which allows computations to be performed on encrypted data, offers a promising solution to these challenges by ensuring that data remains unreadable even if intercepted during computation. This integration is crucial in mitigating privacy risks in federated learning, forming a key discussion point in the subsequent sections of this thesis, which explore the application of homomorphic encryption in FL.

## 2.2 Internet of Things Overview

The Internet of Things (IoT) refers to the expansive network of physical objects that are embedded with sensors, software, and other technologies, enabling them to connect and exchange data with other devices and systems over the internet [51]. These devices range from household items like refrigerators and washing machines to advanced industrial machines [51]. What distinguishes IoT is its ability to gather, share, and analyze vast amounts of data, which can then be leveraged to enhance efficiency and effectiveness in a wide array of applications [2, 23].

However, the rapid proliferation of IoT devices brings significant challenges, particularly related to security and privacy. The **large scale** of IoT networks means that each connected device could

serve as a potential entry point for security breaches. This situation is further complicated by the **heterogeneity** of IoT devices, which often vary widely in terms of their computational power, security capabilities, and operating environments. The **reliability** of these devices can also be unpredictable, leading to vulnerabilities that can be exploited by malicious actors. Consequently, ensuring data integrity and privacy across such a diverse and decentralized network necessitates robust and adaptable security protocols.

As IoT continues to expand, its integration with other emerging technologies like artificial intelligence (AI) and machine learning is making it a critical component in driving the next wave of technological innovation, efficiency, and connectivity. This convergence is not only enhancing the capabilities of IoT systems but also raising new questions about privacy protection and data security. Ensuring that these interconnected devices can operate securely in a wide range of environments is essential for the sustainable growth of IoT, particularly in scenarios where sensitive data is involved [46].

## 2.3 Homomorphic Encryption Preliminaries

This section introduces the fundamental concepts and frameworks underpinning homomorphic encryption, a pivotal technology for enabling secure computations on encrypted data. We begin by defining homomorphic encryption and explaining its significance in privacy-preserving technologies. Following this, we explore different types of homomorphic encryption, including Partially Homomorphic Encryption (PHE) and Fully Homomorphic Encryption (FHE), highlighting their respective functionalities and applications. We then discuss homomorphic encryption schemes based on the Ring-Learning-With-Errors problem, emphasizing their security and practical efficiency. Finally, we conclude with an examination of the security levels associated with homomorphic encryption schemes, particularly in the context of modern cryptographic challenges.

### 2.3.1 Definition

Homomorphic Encryption (HE) is a cryptographic method that permits computations to be carried out on ciphertexts, yielding an encrypted result that, when decrypted, mirrors the outcome of operations as if they were performed on the plaintext data [24]. This unique feature is particularly valuable for performing secure computations on encrypted data, ensuring that privacy is maintained in applications such as cloud computing. HE plays a crucial role in privacy-preserving technologies, including secure data sharing and federated learning.

An encryption scheme $E(k,x)$ for a key $k$ and input $x$ is defined as homomorphic if, for the encryption algorithm $E$ and operation $f$, there exists an efficient algorithm $G$ such that [41]:

$$E(k, f(x_1, \cdots, x_n)) = G(k, f(E(x_1), \cdots, E(x_n))). \tag{2.1}$$

When Equation 2.1 is valid for either addition or multiplication alone, the encryption scheme is referred to as partially homomorphic encryption (PHE). In contrast, if the equation holds for both operations, the scheme is known as fully homomorphic encryption (FHE).

### 2.3.2 Partially Homomorphic Encryption

Partially Homomorphic Encryption supports either addition or multiplication operations on encrypted data, but not both. This limitation makes PHE simpler and faster compared to more general approaches. A classic example of PHE is the Paillier cryptosystem, which supports unlimited additions of ciphertexts and thus is additive homomorphic [52].

The Paillier cryptosystem involves two main processes: encryption and decryption. Given a public key $(n, g)$ and a private key $\lambda$, the system works as follows:

**Encryption:** Given a plaintext message $m$, the ciphertext $c$ is computed as:

$$c \equiv g^m \cdot r^n \ mod \ n^2$$

where $r$ is a random number less than $n$.

**Decryption:** Given a ciphertext $c$, the plaintext $m$ is retrieved by computing:

$$m \equiv \frac{\left(c^\lambda mod n^2\right) - 1}{n} \cdot \mu \ mod \ n$$

where $\mu$ is the multiplicative inverse of $n$ modulo $\lambda$.

The Paillier cryptosystem's key homomorphic property allows two encrypted numbers to be added together to yield the encryption of their sum. Mathematically, if we have two plaintexts $m_1$ and $m_2$, and their respective ciphertexts are $c_1$ and $c_2$ under the same public key, then the encryption of $m_1 + m_2$ is given by:

$$c_{1+2} \equiv c_1 \cdot c_2 \ mod \ n^2$$

Decrypting $c_{1+2}$ will yield $m_1 + m_2$. This property enables secure, encrypted computations that can be performed without needing to reveal the plaintext values.

### 2.3.3 Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) supports both additive and multiplicative operations on ciphertexts, which permits performing any computation on encrypted data without first decrypting it [25]. The development of FHE was a significant breakthrough in cryptography, first achieved by Craig Gentry in 2009 [25]. FHE schemes are constructed to support a theoretically unlimited number of operations without decryption, making them suitable for complex data processing tasks in secure environments.

Homomorphic encryption schemes leveraging the Ring-Learning-With-Errors (RLWE) problem are increasingly favored for their robust security and practical performance. The RLWE problem, a variant of the Learning With Errors (LWE) problem, involves solving systems of linear equations whose coefficients are polynomials in a polynomial ring, typically $R = \mathbb{Z}[x]/(x^n + 1)$, and the errors added to these equations are small polynomials from the same ring [40].

The RLWE problem can be formulated as follows: Given a pair $(a, b)$ where $b \equiv a \cdot s + e \; mod \; q$ and $a$ is chosen uniformly at random from the ring $R_q = R/qR$, $s$ is a secret polynomial, and $e$ is a small error polyomial, the problem is to recover $s$ given $(a, b)$ [40]. The security of RLWE-based schemes relies on the difficulty of this problem, which is believed to be hard, especially against quantum attacks.

The Cheon-Kim-Kim-Song (CKKS) scheme is a prominent example of a HE scheme based on the RLWE problem that supports approximate arithmetic on encrypted data [13]. Unlike other HE schemes that deal with integers, CKKS allows for operations on complex or real numbers, making it especially useful for applications involving floating-point calculations, such as scientific computations and data analytics [13].

In the CKKS scheme, plaintexts are first encoded into polynomials in the ring $R$. The encoding is designed such that small decryption errors (introduced by the encryption and homomorphic operations) do not significantly affect the final decoded result, allowing for approximate but highly practical computations. The encryption process involves $Encrypt(m) : (c_0, c_1) = (p_0 + u \cdot m + e_0, p_1 + e_1)$ where $m$ is the encoded message, $p_0, p_1$ are polynomials from the public key, $u$ are random polynomials for the randomness of ciphertexts, and $e_0, e_1$ are small error polynomials.

CKKS supports both addition and multiplication on ciphertexts:

- Addition: $Add(c, c') = (c_0 + c'_0, c_1 + c'_1)$.
- Multiplication: $Multiply(c, c') = (c_0 \cdot c'_0, c_0 \cdot c'_1 + c_1 + c'_0, c1 \cdot c'_1)$.

The multiplication results in an increase in the degree of the ciphertext, necessitating a procedure known as "relinearization" to reduce it back to a manageable size.

The security of the CKKS scheme, like other RLWE-based schemes, hinges on the hardness of the RLWE problem. It is particularly noted for its capability to handle complex-number computations securely and efficiently, making it ideal for fields that require significant computational precision and security, such as encrypted data analysis and secure machine learning algorithms [13, 41, 48].

### 2.3.4 Security Level

The Security Level of an HE scheme is fundamental in safeguarding data against adversarial entities. Security is typically quantified based on the computational effort (e.g., measured in bits) required to break the encryption. For example, a security level of 128 bits implies that the most effective known attack would necessitate around $2^{128}$ operations to succeed, rendering it impractical with existing and foreseeable technology. The security of HE schemes like those based on the RLWE problem depends heavily on the hardness of the underlying lattice problems, which are believed to be secure against both classical and quantum computing attacks [8].

# Chapter 3

# Design

This chapter details the design decisions and methodologies applied in this project, emphasizing multiparty homomorphic encryption mechanisms that safeguard data privacy and security in a federated learning environment.

We begin by defining the threat model that informs our approach, outlining assumptions and potential adversaries, establishing the security context for our design choices. Next, we examine the multiparty homomorphic encryption schemes central to our privacy-preserving framework, discussing their technical aspects, including the RLWE problem as a secure encryption foundation. We compare different multiparty HE approaches—Plain, Fully, and Threshold Multiparty Homomorphic Encryption—evaluating their strengths, weaknesses, and suitability for various federated learning scenarios.

The chapter then explores the integration of these encryption techniques into federated learning, comparing the efficacy and efficiency of various FL schemes that utilize HE. Finally, we introduce the Hierarchical Multiparty Homomorphic Encryption (H-MHE) protocol, designed to enhance scalability and security in federated learning systems. We describe H-MHE's hierarchical structure, its RLWE-based implementation, and its application in a three-tier federated learning framework, addressing challenges like dropout resilience, system overhead, and communication efficiency to ensure robust, scalable privacy protection.

## 3.1 Threat Model

Follow the threat model of [41], in our work, we apply HE in a HFL scenario to protect data privacy. In this context, we assume that the server, all gateways, and all remotely participating devices are honest-but-curious (zero-trust on privacy). This means that they follow the scheme honestly but

intend to infer the private information of other devices from the information shared during the execution of the protocol. We further assume that collusion may exist between compromised devices and the server [41].

## 3.2 Multiparty Homomorphic Encryption

In this section, we will explore various Multiparty Homomorphic Encryption (MHE) schemes. We use an RLWE-based approach as an illustrative example, in which the ciphertext space is defined within a polynomial quotient ring $R_q = \mathbb{Z}q[X]/(X^n+1)$, where the polynomial degree $n$ is a power of two, and the modulus $q$ for the polynomial coefficients is a product of $L$ distinct primes $q_1, \ldots, q_L$. Consequently, we can leverage the isomorphism $R_q \cong Rq_1 \times \ldots \times R_{q_L}$ provided by the Chinese Remainder Theorem (CRT) to carry out operations in the residue rings, thus avoiding the need for arbitrary-precision integer arithmetic. Additionally, we select each $q_i$ such that $q_i \equiv 1 \ mod \ 2n$, allowing us to represent elements of $R_{q_i}$ in the Number-Theoretic Transform (NTT) domain [49], where both ring operations are performed coefficient-wise. The notation $\alpha \leftarrow \mathbf{D}$ represents the sampling of $\alpha$ according to a distribution $\mathbf{D}$. For the case of uniform sampling of a ring element, we simplify this notation to $\alpha \leftarrow R_q$. Let $\mathbf{Key}(R_q)$ denote a secret-key distribution over $R_q$, where the coefficients are uniformly sampled from $\{-1,0,1\} \ mod \ q$. Let $\mathbf{Err}(R_q)$ refer to an error distribution with coefficients drawn from a discrete Gaussian distribution with small variance $\sigma^2$. Additionally, $\mathbf{Smudge}(R_q)$ refers to a suitable smudging distribution used in noise flooding techniques, typically characterized by a discrete Gaussian distribution with large variance [4, 49]. Lastly, let $\mathbf{CRS}(R_q)$ denote the uniform distribution in $R_q$ according to the common reference string, meaning elements sampled from this distribution are consistent across all parties.

### 3.2.1 Plain Multiparty Homomorphic Encryption

Plain Multiparty Homomorphic Encryption scheme assumes a third-party key generator that issues a common key pair to all data contributors. Each contributor uses this key pair for encryption and decryption [26], shown as Figure 3.1. While simple and low in overhead during the setup, it relies on a weak threat model where a single client colluding with the server can compromise data privacy.

We can write the process of PMHE as a tuple: PMHE = (Setup, SecKeyGen, PubKeyGen, Encrypt, Decrypt).

- PMHE.Setup: A third party agrees on the public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$.
- PMHE.SecKeyGen: The third party samples $S \leftarrow \mathbf{Key}(R_q)$.
- PMHE.PubKeyGen($s$): The third party samples $p_1 \leftarrow \mathbf{CRS}(R_q)$, $e \leftarrow \mathbf{Err}(R_q)$, computes $p_0 =$

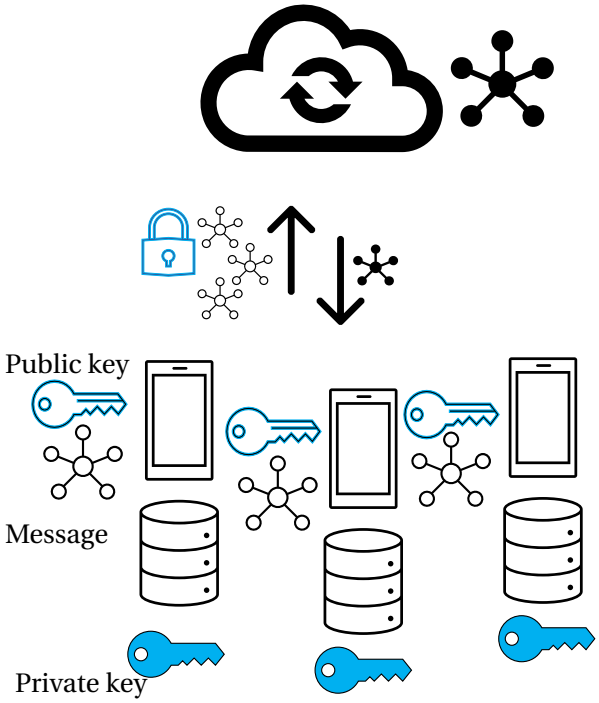


Figure 3.1: An example of PMHE on Federated Learning application.

$-S \cdot p_1 + e$, and sets $pk = (p_0, p_1)$. The third party discloses the key pair $(S, pk)$ to all parties in P.

- PMHE.Encrypt$(pk, m)$: Sample $u \leftarrow \mathbf{Key}(R_q)$, $e_0, e_1 \leftarrow \mathbf{Err}(R_q)$ and output: $ct = (c_0, c_1) = (m + u \cdot p_0 + e_0, u \cdot p_1 + e_1)$.
- PMHE.Decrypt$(ct, s)$: An active party computes $m = c_0 + c_1 \cdot S$ and disclose $m$ to other parties.

### 3.2.2 Fully-Multiparty Homomorphic Encryption

In Fully-Multiparty Homomorphic Encryption scheme, each client holds a local private key, which is a secret share of an unrevealed global private key [49]. All contributors collectively generate the global public key and use it for encryption [41]. Decryption requires cooperation among all parties using their local private keys[41, 49], shown as Figure 3.2. This method has a robust threat model where less than $N-1$ colluders cannot break the cryptosystem. However, it suffers from several drawbacks: a single dropout causes system failure, it is sensitive to the slowest contributor's delay, and it introduces high time overhead and communication costs during setup. Additionally, any configuration update (e.g., adding, removing, or updating a contributor) necessitates resetting the entire cryptosystem.

The FMHE scheme can also be written as a tuple: FMHE = (Setup, SecKeyGen, PubKeyGen,

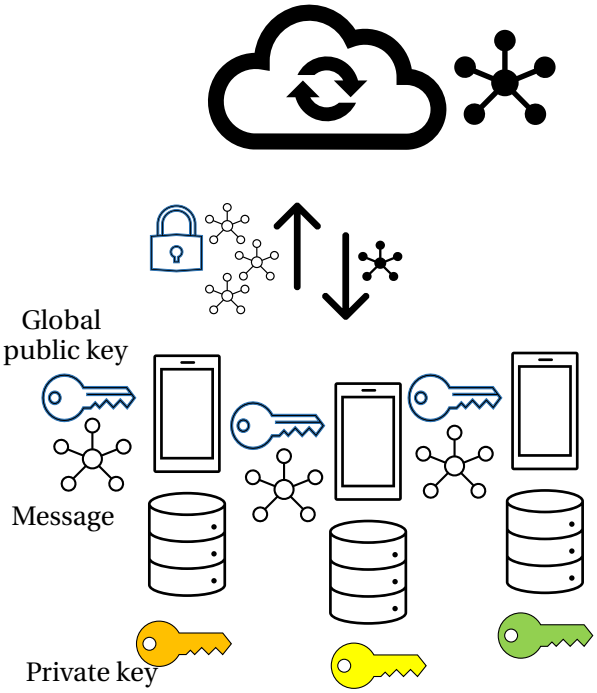


Figure 3.2: An example of FMHE on Federated Learning application.

Encrypt, Decrypt).

- FMHE.Setup: The parties agree on the public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$.
- FMHE.SecKeyGen: Each party $P_i$ samples $s_i \leftarrow \mathbf{Key}(R_q)$.
- FMHE.PubKeyGen$(s_1, \cdots, s_N)$:
  1. Each party $P_i$ samples $p_1 \leftarrow \mathbf{CRS}(R_q)$, $e \leftarrow \mathbf{Err}(R_q)$, and discloses $p_{0,i} = -s_i \cdot p_1 + e$.
  2. Each party computes $p_0 = \sum p_{0,i}$ and sets $pk = (p_0, p_1)$
- FMHE.Encrypt$(pk, m)$: Each party samples $u \leftarrow \mathbf{Key}(R_q)$, $e_0, e_1 \leftarrow \mathbf{Err}(R_q)$ and output: $ct = (c_0, c_1) = (m + u \cdot p_0 + e_0, u \cdot p_1 + e_1)$.
- FMHE.Decrypt$(ct, s)$:
  1. Each party $P_i$ samples $e_i \leftarrow \mathbf{Smudge}(R_q)$ and discloses $h_i = c_1 \cdot s_i + e_i$.
  2. Each party computes $m \simeq c_0 + \sum h_i$

  An active party Pi computes $m = c_0 + c_1 \cdot S$ and disclose $m$ to other parties.

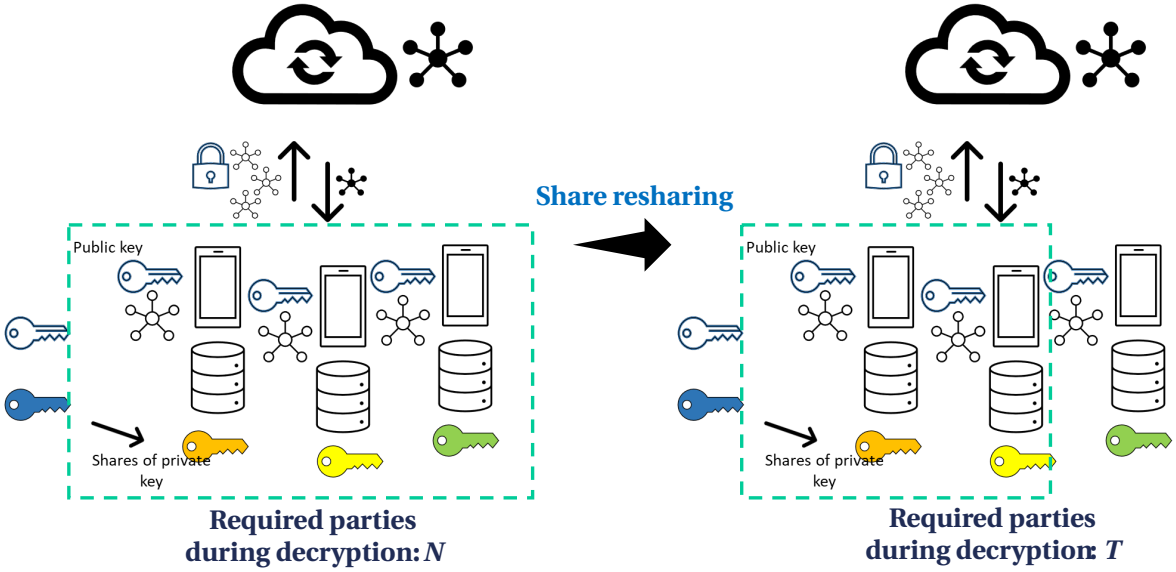


Figure 3.3: Using share-resharing technique to transfer FMHE to TMHE with threshold $T$ out of $N$.

### 3.2.3 Threshold Multiparty Homomorphic Encryption

Threshold-Multiparty Homomorphic Encryption (TMHE) approach, which is developed from FMHE, uses share-resharing techniques, allowing any $T$ of $N$ contributors to decrypt the data ($T < N$), thereby enhancing resilience to dropout [48], shown as Figure 3.3. By adjusting the threshold $T$, it ensures a configurable level of privacy guarantee against collusion attacks. However, while it mitigates the dropout resilience issue, it does not address the overhead introduced during setup as FMHE. Moreover, because of the share-resharing step, the time and communication overhead during setup is significantly increased, especially when the network scale is large.

**Share-resharing** For a set of parties $\mathbb{P}$ in the FMHE scheme where $P_i \in \mathbb{P}$ holds secret-key share $s_i$, the procedure of share-resharing (SR) can be defined as a tuple: SR=(Setup, Thresholdize, Combine) [48]. Intuitively, SR applies the Shamir secret-sharing scheme over $R_q$ to the parties' key, which relaxes the $N$-out-of-$N$ access-structure of FMHE scheme to a $t$-out-of-$N$ threshold one [16, 48, 57].

- SR.Setup: Each party $P_i \in \mathbb{P}$ is associated with a public point $\alpha_i \in R_q$ such that $\alpha_i - \alpha_j$ is a unit for all $i, j, i \neq j$.
- SR.Thresholdize$(t, s_1, \cdots, s_N, \alpha_1, \cdots, \alpha_N)$:
  1. Each party $P_i$ samples $c_{i,1}, \cdots, c_{i,t-1} \leftarrow R_q$.
  2. Each party $P_i$ sends $\tilde{s}_{i,j} = s_i + \sum_{k=1}^{t-1} c_{i,k} \cdot \alpha_j^k$ to each party $P_i$.

3. Each party $P_i$ receives $\tilde{s}_{j,i}$ from each party $P_j$ and computes $\tilde{s}_i = \sum_{j=1}^{N} \tilde{s}_{j,i}$.

- SR.Combine$(\tilde{s}_1, \cdots, \tilde{s}_t, \alpha_1, \cdots, \alpha_t)$: For $\mathbb{P}'$ is a fraction of $\mathbb{P}$, $|\mathbb{P}'| \geq t$, each party $P_i \in \mathbb{P}'$ computes $s'_i = \tilde{s}_i \prod_{j=1, i\neq j}^{t} (\alpha_j / (\alpha_j - \alpha_i))$.

By combining FMHE and SR, we can derive TMHE as the union tuple FMHE $\cup$ SR [47]:

- TMHE.Setup: run the FMHE.Setup and SR.Setup steps.
- TMHE.SecKeyGen:
  1. execute $(s_1, \cdots, s_N) \leftarrow$ MHE.SecKeyGen.
  2. run SR.Thresholdize$(t, s_1, \cdots, s_N, \alpha_1, \cdots, \alpha_N)$.
- TMHE.PubKeyGen$(t, \tilde{s}_1, \cdots, \tilde{s}_t)$:
  1. retrieve $\mathbb{P}_{online}$ from the environment, selecting the online parties.
  2. if $|\mathbb{P}_{online}| < t$, abort this attempt
  3. choose $t$ parties in $\mathbb{P}_{online}$ and run $(s'_1, \cdots, s'_t) \leftarrow$ SR.Combine
  4. run FMHE.PubKeyGen$(s'_1, \cdots, s'_t)$
- TMHE.Encrypt$(pk, m)$: execute FMHE.Encrypt
- TMHE.Decrypt$(ct, \tilde{s}_1, \cdots, \tilde{s}_t)$
  1. retrieve $\mathbb{P}_{online}$ from the environment, selecting the online parties.
  2. if $|\mathbb{P}_{online}| < t$, abort this attempt
  3. choose $t$ parties in $\mathbb{P}_{online}$ and run $(s'_1, \cdots, s'_t) \leftarrow SR.Combine$
  4. conduct FMHE.Decrypt$(ct, s'_1, \cdots, s'_t)$ procedure.

## 3.3 FL with HE

In the realm of privacy-preserving federated learning, integrating HE has emerged as a pivotal strategy to ensure data confidentiality during the training process. The integration schemes can be broadly classified into three categories, depending on the type of HE system utilized: 1) FL with PMHE; 2) FL with FMHE; 3) FL with TMHE.

**FL with PMHE** FL with PMHE focuses on applying basic HE techniques to secure the model training process. In these schemes, the participants encrypt their model updates using HE under the same key pair before sending them to the central aggregator. The aggregator then performs encrypted aggregation and sends the encrypted model back to the participants [26]. This approach, while enhancing privacy and simple, poses severe disadvantages. First, such schemes usually need a third party to generate and distribute the key pair, which is infeasible in a zero-trust scenario. Second, these schemes are extremely weak to single-point failure. For internal attackers, the server can collude with a client to obtain the common private key. For external attackers, the adversary can derive the common private key from insecure communication channels and then decrypt any encrypted model updates he can access. Research in this area aims at optimizing encryption schemes to make them more practical for large-scale FL scenarios where emphasize on efficiency but less on privacy [26], or cross-silo FL frameworks where data centers (clients) are highly reliable [71]. A typical FedAVG with PMHE process is shown as Algorithm 1.

**FL with FMHE** FL with FMHE addresses the limitations of PMHE by allowing multiple participants to encrypt their data with individual keys while still enabling secure aggregation on the encrypted data [41, 49]. This approach facilitates a more flexible and secure FL framework, as it supports direct computations on encrypted data from different participants without requiring decryption. And the requirement of all-party-participation during decryption also ensures high guarantee against collusion. On the other hand, requiring all parties to contribute also introduces high overhead when the scale of clients is large. Also, it provides no resilience to delay or dropout of client devices, so it's infeasible to adapt MK-HE directly to unreliable and heterogeneous IoT networks. MK-HE is particularly useful in collaborative FL settings where data privacy among multiple data holders is paramount.

**FL with TMHE** FL with TMHE incorporates a threshold-based decryption mechanism, enhancing security and trust in federated learning environments. In these schemes, decryption of the aggregated model requires a minimum number of participants to combine their decryption shares [18, 49]. This method not only secures the aggregation process but also adds an additional layer of control over the model's access, making it suitable for highly sensitive applications. The threshold mechanism ensures certain guarantee against collusion while preserve the resilience to delay and dropout. Nonetheless, the thresholdization process introduce large over head compared to FMHE. And when the threshold is large, the system could suffer from high overhead and latency, which is a big problem in cross-device FL scenarios. And when the threshold is small, the adversary can easily corrupt the system by collusion. So a good trade-off shall be made by adjusting the threshold. TMHE-based FL schemes are still under exploration, with research focusing on their feasibility and efficiency in real-world applications.

The existing MHE schemes in federated learning, while offering various advantages in preserving privacy, face significant limitations:

**Algorithm 1** FedAvg with PMHE

**Input:** Number of rounds $T$, number of clients $K$, number of local epochs $E$, batch size $B$, learning rate $\eta$

---

**PKI Initialization:**
Generate HE key pair $(pk, sk)$
Distribute public key $pk$ to the server and all clients
Distribute private key $sk$ to all clients

---

**Server Side:**
**for** $t = 1$ **to** $T$ **do**
  Server selects a subset of clients $\mathbb{S}_t$ randomly
  **for all** selected clients $k \in \mathbb{S}_t$ **in parallel do**
    Server sends encrypted global aggregated model weights $\text{Enc}_{pk}(|\mathbb{S}_{t-1}| \cdot w_t)$ and the number of participants in the global model $|\mathbb{S}_{t-1}|$ to client $k$
  **end for**
  **for all** selected clients $k \in \mathbb{S}_t$ **in parallel do**
    Client $k$ computes encrypted updates $\text{Enc}_{pk}(w_{t+1}^k)$ using local data
    Client $k$ sends encrypted updates back to the server
  **end for**
  Server computes encrypted aggregated update: $\text{Enc}_{pk}(|\mathbb{S}_t| \cdot w_{t+1}) = \sum_{k \in \mathbb{S}_t} \text{Enc}_{pk}(w_{t+1}^k)$
**end for**

---

**Client Side ($k$):**
**Require:** Encrypted global model weights $\text{Enc}_{pk}(|\mathbb{S}_{t-1}| \cdot w_t)$, the number of participants in the global model in the last round $|\mathbb{S}_{t-1}|$, local data, HE key pair $(pk, sk)$, number of participants in the global aggregated model $|\mathbb{S}_{t-1}|$
Client $k$ decrypts the aggregated updated model and gets the average: $w_t = \frac{1}{|\mathbb{S}_{t-1}|}\text{Dec}_{sk}(\text{Enc}_{pk}(|\mathbb{S}_{t-1}| \cdot w_t))$
**for** $e = 1$ **to** $E$ **do**
  **for** batch $b \in$ local data **do**
    Compute gradient $\nabla L(w_t, b)$ on batch $b$
    Update weights: $w_t = w_t - \eta \nabla L(w_t, b)$
  **end for**
**end for**
Compute encrypted update: $\text{Enc}_{pk}(w_{t+1}^k)$
Return encrypted update to the server

- FL with PMHE is vulnerable to single-point failures and internal or external attacks due to the reliance on a common private key. The necessity of a third-party key distribution in these schemes is impractical in zero-trust environments, limiting their applicability in highly secure scenarios.
- FL with FMHE improves security by allowing participants to use individual keys, but it introduces substantial overhead, particularly in large-scale settings. The requirement for all-party participation in decryption makes it unsuitable for unreliable networks, such as those in IoT environments, due to its lack of resilience to client delays or dropouts.
- FL with TMHE offers enhanced security through threshold-based decryption but suffers from high overhead and latency during system setup, making it less practical in cross-device FL scenarios. A large threshold will also introduce large overhead, while a small threshold can compromise security, as it becomes easier for adversaries to corrupt the system through collusion. Thus, balancing the threshold is crucial but challenging.

These limitations highlight the need for research to optimize these schemes for practical and efficient use in federated learning across diverse environments.

## 3.4 Hierarchical Multiparty Homomorphic Encryption Protocol with RLWE-Based Scheme

In the evolving landscape of Federated Learning, the paramount importance of data privacy stands as a critical challenge, particularly in the context of Internet of Things (IoT) applications. The burgeoning IoT ecosystem, characterized by its vast scale, device heterogeneity, and the inherent unreliability of network connections, presents unique challenges that demand innovative solutions.

The adaptation of Homomorphic Encryption to the FL paradigm has been identified as a promising avenue to address these privacy concerns. However, existing MHE schemes, predominantly designed with cross-silo FL or reliable networks in mind, are not directly transferrable to or optimized for the intricate requirements of IoT scenarios. This disconnect between existing HE solutions and the practical demands of IoT-based FL highlights a significant gap in the current research and application landscape.

To address this gap, we propose a novel Hierarchical Homomorphic Encryption (H-MHE) protocol specifically designed for Hierarchical Federated Learning (HFL), which extends existing MHE schemes to a hierarchical and hybrid access structure. This innovative approach seeks to tailor the privacy-preserving capabilities of HE to the nuanced dynamics of IoT environments. By conceptualizing a three-tier FL framework that integrates PMHE, FMHE, and TMHE, our proposal aims to create a robust and practical solution for securing cross-device FL, e.g., IoT applications.

The H-MHE protocol is engineered to leverage the strengths of each HE type, thereby ensuring comprehensive privacy protection across different layers of the FL hierarchy.

From the discussion above, we found that MHE protocols enable the system to function as an abstract encryption and decryption machine using a virtual public-private key pair (e.g., PMHE, FMHE, TMHE). The core idea of our proposal, the Hierarchical Multiparty Homomorphic Encryption (H-MHE) framework, extends these abstracted machines into a hierarchical structure. The H-MHE framework is inherently scalable to any number of layers, as it is also an MHE system. To illustrate, a three-tier H-MHE system (with one layer of abstraction) can be adapted into a three-tier Hierarchical Federated Learning (HFL) system for privacy-preserving machine learning. HFL systems, compared to traditional two-tier server-client Federated Learning frameworks, introduce additional layers between the central server and clients for intermediate aggregation.

Here we propose a feasible design for the H-MHE framework based on Ring-Learning-With-Error (RLWE) encryption scheme. This is a three-tier H-MHE framework, in which client devices are divided into several clusters during setup phase. Each cluster contains a gateway. The gateway can be seen as an assistant sub-server for the cluster and is also the representative of the cluster to communicate with other clusters. Each cluster runs an MHE protocol. Among the gateways of all clusters also runs an abstract MHE protocol (can be with the assistance of a central server). All MHE protocols run on the same public parameter setting and a public common random string (CRS).

In our design, each cluster runs a protocol $\psi \in \{\text{PMHE, FMHE, TMHE}\}$, and among gateways (i.e., abstract devices) and the central server we run abstract FMHE protocol. To describe H-MHE design, clusters with indices in set $\mathbb{D}$ run FMHE, clusters with indices in set $\mathbb{V}$ run TMHE, and clusters with indices in set $\mathbb{L}$ run PMHE. We define this H-MHE protocol as the five-tuple of procedures H-MHE = (Setup, SecKeyGen, PubKeyGen, Encrypt, Decrypt):

- H-MHE.Setup: Each cluster $C_d$, $d \in \mathbb{D}$ runs FMHE.Setup, each cluster $C_v$, $v \in \mathbb{V}$ runs TMHE.Setup, and each cluster $C_l$, $l \in \mathbb{L}$ runs PMHE.Setup, on the public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$.
- H-MHE.SecKeyGen: Each cluster $C_d$, $d \in \mathbb{D}$ runs $d := \{s_{d,1}, \ldots, s_{d,N_d}\} \leftarrow$ FMHE.SecKeyGen, each cluster $C_v$, $v \in \mathbb{V}$ runs $S_v := \{s_{v,1}, ..., s_{v,N_v}\} \leftarrow$ TMHE.SecKeyGen, and each cluster $C_l$ running PMHE runs $S_l \leftarrow$ PMHE.SecKeyGen.
- H-MHE.PubKeyGen$(S_i, i \in \mathbb{D} \cup \mathbb{V} \cup \mathbb{L})$:
  - Each cluster $C_d$, $d \in \mathbb{D}$ runs $(p_{0,d}, p_1) \leftarrow$ FMHE.PubKeyGen$(S_d)$, each cluster $C_v$, $v \in \mathbb{V}$ runs $(p_{0,v}, p_1) \leftarrow$ TMHE.PubKeyGen$(S_v)$, and each cluster $C_l$, $l \in \mathbb{L}$ runs $(p_{0,l}, p_1) \leftarrow$ TMHE.PubKeyGen$(S_l)$. Each cluster discloses its $p_{0,I}$ to other clusters.
  - Each cluster computes $p_0 = \sum p_{0,I}$ and sets $pk = (p_0, p_1)$.
- H-MHE.Encrypt$(pk, m)$: run the FMHE.Encrypt procedure.
- H-MHE.Decrypt$(ct, S_i, i \in \mathbb{D} \cup \mathbb{V} \cup \mathbb{L})$:

– Parallelly each cluster runs:
  * For each cluster $C_d$, $d \in \mathbb{D}$, each party $P_{d,i} \in C_d$ samples $e_{d,I} \leftarrow$ **Smudge**$(R_q)$ and discloses $h_{d,I} = c_1 \cdot s_{d,i} + e_{d,i}$.
  * For each cluster $C_l$, $l \in \mathbb{L}$, pick up an online party $P_{l,i} \in C_l$. $P_{l,i}$ samples $e_{l,I} \leftarrow$ **Smudge**$(R_q)$ and discloses $h_{l,I} = c_1 \cdot S_l + e_{l,i}$.
  * For each cluster $C_v$, $v \in \mathbb{V}$:
    · Obtain $P_{online} \leftarrow$ Env (pick online parties from the environment)
    · If $|P_{online}| < t_v$, return `Null`
    · Choose $t_v$ parties $P_{online}$ and run $(s'_{v,1}, \cdots, s'_{v,t_v}) \leftarrow$ T.Combine
    · each party $P_{v,i}$ in $P_{online}$ samples $e_{v,I} \leftarrow$ **Smudge**$(R_q)$ and discloses $h_{v,I} = c_1 \cdot s'_{v,i} + e_{v,i}$.
    · Each cluster can then compute $m \simeq c_0 + \sum h_{i,j}$.

## 3.5 Integrating HFL with H-MHE

The three-tier H-MHE protocol can be adapted to the three-tier hierarchical federated learning framework. A representative framework is that in H-MHE procedure, each party performs a round of local training before H-MHE.Encrypt. During H-MHE.Encrypt, each party encrypts its updated model using the global public key. Then each party sends the encrypted updated model to the gateway in its cluster. Each gateway adds all received messages up (homomorphic addition) and send the aggregated message to the central server. The central server then adds all received messages up (homomorphic addition) and disclose the aggregated message to all clusters for H-MHE.Decrypt and derive the plaintext of aggregated model. The framework is shown as Figure 3.4.

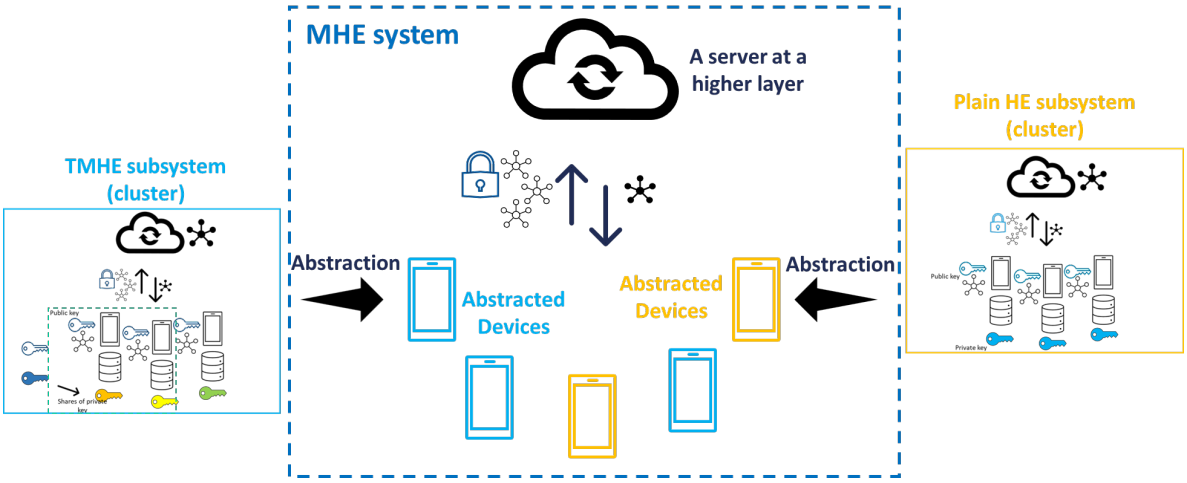


Figure 3.4: Design of H-MHE on Federated Learning application.

# Chapter 4

# Implementation and Evaluation

This chapter details the implementation and evaluation of the Hierarchical Multiparty Homomorphic Encryption (H-MHE) framework integrated with Federated Learning (FL). It covers key aspects of the project's implementation, focusing on practical considerations, tools used, and a comprehensive evaluation of the framework's performance in various scenarios.

We begin by introducing the programming environment and key third-party libraries used. The Go programming language was chosen for its concurrency support and performance advantages, particularly in multiparty computation simulations. We also utilized the `Lattigo` library for lattice-based cryptography and the `go-deep` library for neural network integration. Following this, we delve into the H-MHE protocol's implementation, outlining the steps taken to adapt homomorphic encryption schemes to a hierarchical federated learning framework. We describe the design, integration of various encryption methods, and the roles of different layers (server, gateway, and client devices) in the hierarchical architecture, ensuring data privacy and security at each level.

The chapter then transitions to the evaluation phase, where we assess the H-MHE framework's performance. We simulate a heterogeneous large-scale network to test the approach's resilience and practicality, particularly in IoT environments. Through various experiments, we evaluate model accuracy, communication and computation overhead, and the framework's ability to tolerate device dropouts. The results demonstrate the advantages of our approach over traditional multiparty homomorphic encryption schemes, highlighting the flexibility, scalability, and privacy guarantees of the H-MHE framework.

## 4.1 Utility Basics

We implement the H-MHE demo using `Go` language, because `Golang` provides good concurrency support like `Gorountines` and `Channels` [17], and good performance with efficient memory management, which is well suitable for multiparty computation simulation. In this section we will introduce two main third-party libraries that we used in our implementation.

### 4.1.1 `Lattigo v5.0.2`: Lattice-Based Multiparty Homomorphic Encryption Library in Go

`Lattigo` is an open-source library written in Go, designed specifically for implementing lattice-based cryptographic protocols [34]. It is built on the Ring-Learning-With-Errors (RLWE) cryptographic assumption, making it suitable for applications requiring security against quantum computing threats [34]. Lattigo supports several advanced schemes, such as the CKKS scheme for approximate arithmetic on encrypted data.

`Lattigo v5.0.2` provides tools for:

- **Key Generation:** Fast and secure generation of public and private keys.
- **Encryption and Decryption:** Efficient methods to encrypt and decrypt messages, maintaining the integrity and confidentiality of data.
- **Homomorphic Operations:** Functions to perform addition, multiplication, and more complex arithmetic on encrypted data without decryption.
- **Multiparty Computation:** `Lattigo` also provides basic multiparty computation features for implementing multiparty homomorphic encryption systems.

`Lattigo` is designed with a focus on performance and usability in Go environments, offering a robust platform for researchers and developers to integrate lattice-based cryptography into their applications. In our implementation, we utilized the CKKS scheme and some of the multiparty computation features to build our H-MHE framework.

### 4.1.2 `go-deep`: Feed Forward/Back-Propagation Neural Network Implementation

`go-deep` is a lightweight, versatile neural network library written in `Go` [19], intended for building and training feedforward and backpropagation neural networks. It supports various activation functions such as Sigmoid, Tanh, and ReLU, making it flexible for different types of neural network applications.

Features of `go-deep` include:

- **Model Definition:** Simple and intuitive API for defining network architecture, including the number of layers, neurons, and activation functions.
- **Training and Evaluation:** Tools for training networks using batch gradient descent, stochastic gradient descent, or other optimization methods. It also provides functionalities for evaluating the performance of the trained models.
- **Scalability:** Although primarily designed for smaller scale applications, go-deep can be scaled up for larger datasets with careful optimization.

`go-deep` is particularly useful for a straightforward neural network implementation in `Go`, without the overhead of more complex and feature-rich machine learning frameworks. Its ease of use and the efficient execution make it suitable for both research purposes and real-world applications in machine learning where `Go` is the preferred environment. In our implementation, we further added HFL features to `go-deep`.

## 4.2 A Privacy-Preserving Federated Learning Demo with Hierarchical Multiparty Homomorphic Encryption Protocol using RLWE-Based Scheme

This section will delve into the implementation of H-MHE, the architectural design of our proposed H-MHE framework, and the methodology we employ in implementing and validating our protocol. Through this comprehensive exploration, this thesis aims to fortify the privacy and security landscape of FL in IoT environments, thus paving the way for more secure, efficient, and reliable machine learning solutions in the age of ubiquitous computing.

### 4.2.1 HFL Framework

In this section we focus on a common three-tier large-scale IoT networks running FL applications with heterogeneous delays and unreliable networks [70], shown as Figure 4.1. The Hierarchical Federated Learning (HFL) framework is designed to accommodate the inherent complexities and dynamics of IoT environments. By organizing the federated learning process into three distinct layers: server, gateways, and client devices, the HFL architecture aims to efficiently manage the vast scale, device heterogeneity, and network variability characteristic of IoT applications. This section outlines the roles and interactions of these three layers within the HFL framework.

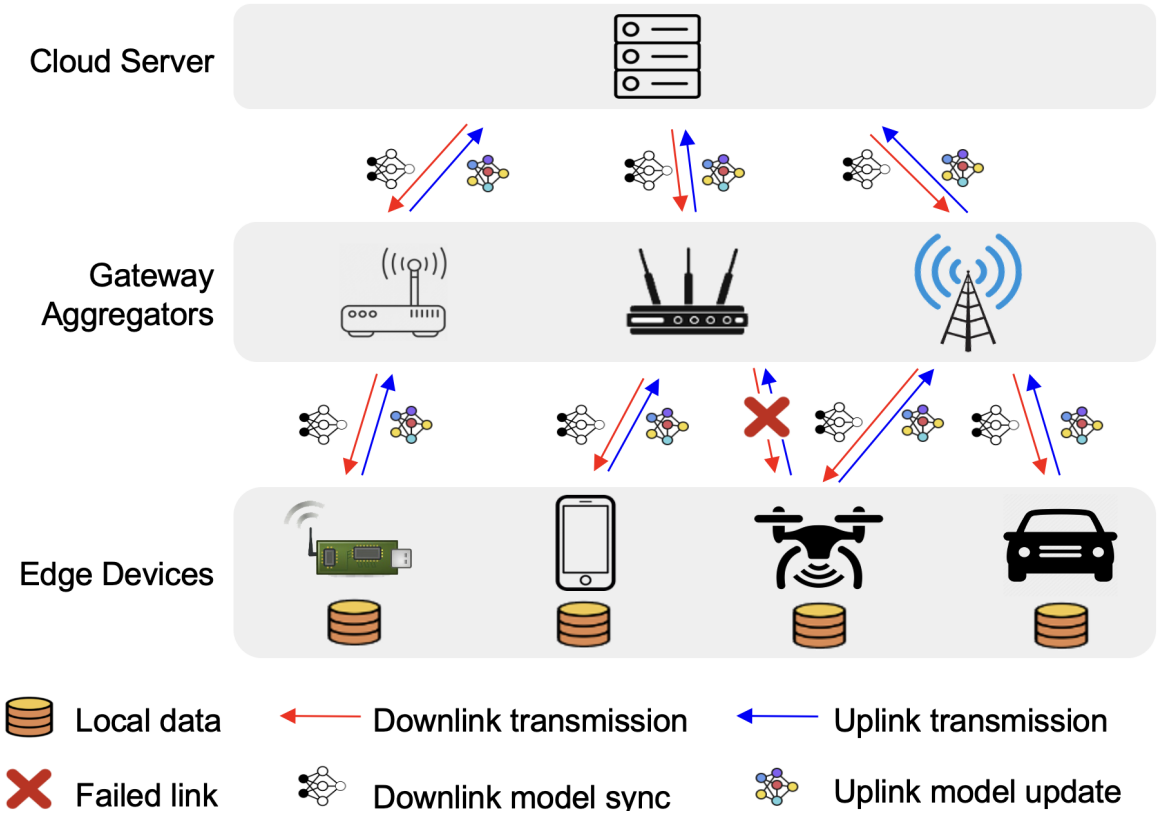


Figure 4.1: System architecture for a three-tier IoT networks running FL applications with heterogeneous delays and unreliable networks [70].

**Server Layer** The server layer serves as the central coordinating entity in the HFL framework. Its primary function is to initiate the federated learning process, aggregate model updates from the gateway layer, and update the global model. The server is responsible for orchestrating the learning process across the entire network, ensuring that model updates are securely aggregated and that the updated global model is disseminated back to the gateways for further distribution.

**Gateway Layer** The gateway layer acts as an intermediary between the server and the client devices. Gateways are strategically positioned within the network to manage communications, data preprocessing, and local aggregations from a subset of client devices. This layer significantly reduces the communication overhead between the server and the vast number of client devices by aggregating local updates before passing them to the server.

**Client Devices Layer** The client devices layer consists of the myriad IoT devices that participate in the learning process by computing model updates based on their local data. These devices

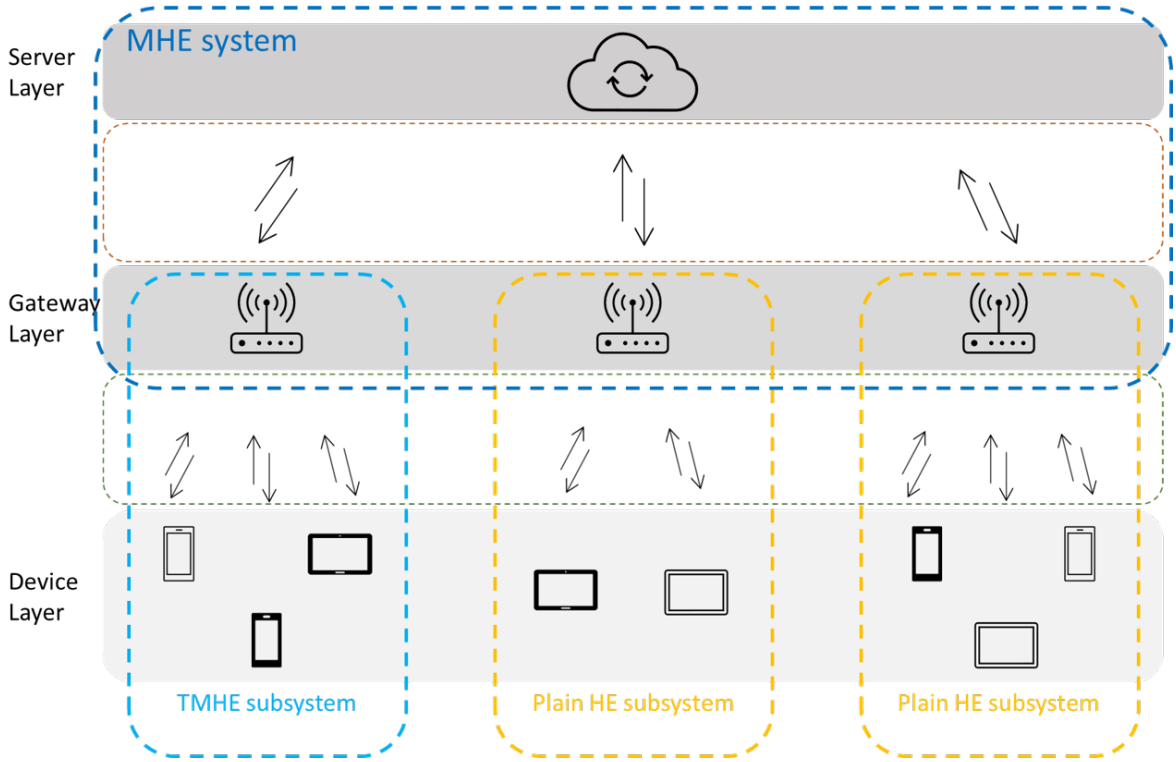


Figure 4.2: Hierarchical Homomorphic Encryption Framework for Hierarchical Federated Learning on IoT Networks.

could range from smartphones and wearable devices to sensors and smart home appliances. Given the resource constraints typical of many IoT devices, the HFL framework is designed to minimize their computational and communication burdens. Client devices perform local computations and transmit their updates to their respective gateways.

The HFL framework's hierarchical structure is tailored to address the scalability, heterogeneity, and privacy challenges of federated learning in IoT environments. By distributing the learning process across these three layers, the HFL architecture achieves efficient resource utilization, reduces communication costs, and enhances privacy preservation, making it particularly well-suited for large-scale, real-world IoT applications.

### 4.2.2 Multiparty Homomorphic Encryption with Hierarchical and Hybrid Access Structure

We discussed PMHE, FMHE and TMHE in Section 3.2, now we adopt these three encryption scheme to the Hierarchical HE framework (H-MHE), shown as Figure 4.2.

Here, the framework follows a cluster-based structure, where client devices are divided by

clusters and each gateway is dedicated to handling one cluster as a proxy. The H-MHE follows a hybrid constructions, **where the TMHE scheme or the PMHE scheme is used to emulate a single key within a FMHE setting** [49]. In other words, the Gateway Layer becomes an abstraction for both the Server Layer and the Device Layer:

- For the Server Layer, gateways are the proxies of client devices, so the server views gateways as "real" parties participating FMHE. Thus, the interaction between the Server Layer and the Gateway Layer is the same as a two-tier FL framework with FMHE. This also provides options for clusters between the Gateway Layer and the Device Layer, that they can choose to use PMHE scheme and use simple key pairs to participate in FMHE, or use TMHE scheme to emulate a single key pair. Despite what scheme clusters use, a gateway always follows FMHE scheme as a single party in the view of the server.
- For the Device Layer, the gateway dedicated to the corresponding cluster is viewed as the "real" server by client devices. And as discussed above, in this cluster, the dedicated gateway and client devices can choose rather to use THE scheme or Plain HE scheme. In key generation and encryption step, TMHE scheme requires client devices to follow the TMHE key generation to collectively generate private key shares held by each device as well as the public key for this cluster [49]. While in PMHE scheme, the gateway or a third party generates the key pair for the cluster and broadcast the public key to its client devices. During decryption, TMHE scheme requires client devices first collectively partially decrypt the ciphertext, and then uses the result to participate the FMHE decryption. On the other hand, in PMHE scheme, the cluster first partially decrypt the ciphertext using its private key, and then use the result for FMHE decryption.

### 4.2.3 Integration of Hierarchical Multiparty Homomorphic Encryption Protocol and Federated Learning

To describe a general Multiparty Homomorphic Encryption (MHE) system, We follow the generic expressions of [49]. Let $\mathbb{C} = \{C_1, C_2, \cdots, C_K\}$ be a set of $K$ clusters and $\mathbb{G} = \{G_1, \cdots, G_K\}$ be the corresponding dedicated gateways. Let $\mathbb{T}$ be the index set of clusters who use TMHE scheme, and $\mathbb{L}$ be the index set of clusters who use PMHE scheme. Thus $\mathbb{T} \cup \mathbb{L} = \{1, 2, \cdots, K\}$ and $\mathbb{T} \cap \mathbb{L} = \emptyset$. The set of client devices in $C_k$ is $\mathbb{D}_k = \{D_1^k, D_2^k, \cdots, D_{N_k}^k\}$, where $N_k$ is the number of devices in $C_k$.

The private key of an $N$-party MHE system is an $N$-party function $\mathbb{S}(sk_1, sk_2, \cdots, sk_N)$. The structure of $\mathbb{S}$ determines the Access Structure of the MHE scheme [49], which we define as the set $S \subset \texttt{PowerSet}(\mathbb{P})$ of all groups of parties that can collectively reconstruct the private key. Indeed, $\mathbb{S}$ should never be disclosed in practice.

**HE Scheme** Privacy-preserving machine learning leveraging HE often utilizes the Cheon-Kim-Kim-Song (CKKS) scheme, an advanced scheme based on Ring-Learning-with-Errors (RLWE) due to its unique ability to efficiently handle arithmetic on encrypted floating-point numbers [10, 12, 21, 30, 41]. Unlike other HE schemes that are limited to operations on integers, CKKS facilitates complex computations necessary for machine learning algorithms, which frequently involve real numbers and require precision in calculations. The CKKS scheme stands out for its approximation techniques that enable it to perform additions, multiplications, and even more complex functions like exponentiation on encrypted data without significant loss in precision. This capability is crucial for maintaining the accuracy of machine learning models while ensuring data privacy. In this work, we will use the multiparty variants of CKKS.

**Trade-off between Privacy and Practicality** In HFL scenarios, communication is expensive, so collective decryption only on the Device Level is infeasible, for it requires devices to communicate with each other frequently, which also introduces high computation overhead. Thus, to adapt H-MHE to HFL, we decide to maintain the partial decryption procedures that only require the participation of one device at the Device Level, but move all computations during learning process that involve several entities to be proxied by the Gateway Level or the Sever Level (e.g., collectively retrieving aggregated model from partially decrypted ciphertexts). This will lead to a vulnerability that the Gateway Level and the Server Level can access the aggregated model, which may expose to adversaries. However, we argue that it is nearly impossible to prevent the leakage of the aggregated model in IoT scenarios, as any attackers can simply register a cheap IoT device (compared with expensive qualified data centers in cross-silo FL) in a legitimate way in the IoT networks to access the aggregated model, even if the model is hidden from the server and gateways during the learning process. Thus, to mitigate Model-based Attacks in cross-device FL, other techniques must be employed, which is out of the context of our work. To make H-MHE practical in IoT scenarios, we argue that it is reasonable to tolerate the risks of exposing the aggregated model to the Server Level and the Gateway Level, for much better practicality. And to reduce the number of communication rounds and to mitigate the risk of Model-based Attacks, we will use FedAVG scheme in the local training of client devices [42].

Now we can derive a practical H-MHE framework on FL. The protocols are shown in Algorithm 3, 4, 5, 6. Apparently, most of difficult multiparty computations happen on the Server Level and the Gateway Level, and the Device Level only handles local training as well as only one encryption operation and at most one partially decryption operation, which mostly leverage the limited performance of IoT devices. We will discuss the security aspect and its benefits on practicality in following sections.

**Algorithm 2** Setup Protocol of H-MHE on HFL

**Input:** Number of clusters $K$, number of devices in cluster $(k)$ $N_k$, $k \in 1, \cdots, K$, index set of clusters who use TMHE scheme $\mathbb{T}$, index set of clusters who use PMHE scheme $\mathbb{L}$.

---

**Server Side:**
Distribute the public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$ to all gateways.
The server receives local public key list $(pk_1, pk_2, \cdots, pk_K)$ from all gateways.
The server aggregates all local public keys and obtains the global public key $\mathbf{P} \leftarrow$ `FMHE.PubKeyGen.step2`$(pk_1, pk_2, \cdots, pk_K)$ and distributes to all gateways.

---

**Gateway Side ($l$, $l \in \mathbb{L}$):**
**Require:** Public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$.
Gateway $G_l$ or a third party generates a key pair $(pk_l, sk_l) \leftarrow$ `CKKS.KeyGen`$(n, q, \chi, \psi, \mathbf{a})$ for its cluster $C_l$. For consistency in expression, we note $sk_l$ as a function $\mathbf{S}_l()$.
Gateway $G_l$ sends the local public key $pk_t$ to the server.
Gateway $G_l$ receives the global public key $\mathbf{P}$ from the server and distributes to all devices in its cluster $C_l$.

---

**Gateway Side ($t$, $l \in \mathbb{T}$):**
**Require:** Public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$.
Gateway $G_t$ distributes the public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$ to the devices in its cluster $C_t$.
Gateway $G_t$ receives the local public key $pk_t$ from devices in the cluster $C_t$.
Gateway $G_t$ sends the local public key $pk_t$ to the server.
Gateway $G_t$ receives the global public key $\mathbf{P}$ from the server and distributes to all devices in its cluster $C_t$.

---

**Device Side ($j$ in Cluster $l$, $l \in \mathbb{L}$):**
Device $D_j^l$ receives the aggregated public key $\mathbf{P}$ from its gateway $G_l$.

---

**Device Side ($j$ in Cluster $t$, $t \in \mathbb{T}$):**
**Require:** Public parameters $(n, q, \sigma, \mathbf{Key}, \mathbf{Err})$, TMHE threshold $\mathbf{T}_t$.
Devices $D_j^t$, $j \in \{1, \cdots, N_t\}$ generate $sk_i^t \leftarrow$ `TMHE.SecKeyGen`$((n, q, \chi, \psi, \mathbf{a}), \mathbf{T}_t)$ and take part in the multiparty protocol inside the cluster $pk_t \leftarrow$ `TMHE.PubKeyGen`$(sk_1^t, \cdots, sk_{N_t}^t)$. Outputs a key pair $(sk_i^t, pk_t)$ to each device in $C_t$. And thus in cluster $C_t$, we have a private key as a function $\mathbf{S}_t(sk_1^t, \cdots, sk_{N_t}^t)$.
Device $D_j^t$ sends local public key $pk_t$ to its gateway $G_t$.
Device $D_j^t$ receives the aggregated public key $\mathbf{P}$ from the gateway $G_t$.

**Algorithm 3** Training Protocol of H-MHE on H-FL (Server Layer)

**Input:** Number of rounds $T$, number of local epochs $E$, batch size $B$, learning rate $\eta$, number of clusters $K$, number of devices in cluster $(k)$ $N_k$, $k \in 1, \cdots, K$, index set of clusters who use TMHE scheme $\mathbb{T}$, index set of clusters who use PMHE scheme $\mathbb{L}$, FMHE global public key $\mathbf{P}$, local public key list $(pk_1, \cdots, pk_K)$

---

**Server Side:**
**for** $i = 1$ **to** $T$ **do**
  **for all** Gateways $k \in 1, \cdots, K$ **in parallel do**
    Server sends global average model weights $\cdot w_i$ to gateway $k$
  **end for**
  **for all** Gateways $k \in 1, \cdots, K$ **in parallel do**
    Gateway $k$ computes partially-aggregated encrypted update $\text{Enc}_{\mathbf{P}}(\Sigma_{j \in |S_i^k|} w_{i+1}^{k,j})$ in its cluster.
    Gateway $k$ sends $\text{Enc}_{\mathbf{P}}(\Sigma_{j \in |S_i^k|} w_{i+1}^{k,j})$ as well as the number of devices submitting their updates in its cluster $|S_i^k|$ back to the server.
  **end for**
  Server computes encrypted fully-aggregated update: $\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1}) = \sum_{k \in 1, \cdots, K} \text{Enc}_{\mathbf{P}}(\Sigma_{j \in |S_i^k|} w_{i+1}^{k,j})$.
  **for all** Gateways $k \in 1, \cdots, K$ **in parallel do**
    Server sends encrypted global aggregated model weights $\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1})$ to gateway $k$
  **end for**
  **for all** Gateways $k \in 1, \cdots, K$ **in parallel do**
    Gateway $k$ computes partially-decrypted global aggregated model weights $\text{Dec}_{\mathbf{S}_k}(\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1}))$ .
    Gateway $k$ sends $\text{Dec}_{\mathbf{S}_k, \mathbf{P}}(\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1}))$ back to the server.
  **end for**
  Server computes plaintext global aggregated model weights by aggregating all partially-decrypted models: $|S_i| \cdot w_{i+1} = \sum_{k \in 1, \cdots, K} \text{Dec}_{\mathbf{S}_k, \mathbf{P}}(\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1}))$.
  Server does an average on the aggregated model and get the global average weights: $w_{t+1} = \frac{1}{|S_i|} \cdot |S_i| \cdot w_{i+1}$
**end for**

**Algorithm 4** Training Protocol of H-MHE on H-FL (Gateway Layer)

**Gateway Side ($l$, $l \in \mathbb{L}$):**
**Require:** global average model weights $\cdot w_i$, FMHE global public key $\mathbf{P}$, local private key $\mathbf{S}_l$.
Gateway selects a subset of devices $S_i^l$ randomly.
**for all** selected devices $j \in S_i^l$ **in parallel do**
  Gateway sends global average model weights $\cdot w_i$ to device $j$.
**end for**
**for all** selected devices $j \in S_i^l$ **in parallel do**
  Device $j$ computes encrypted updates $\text{Enc}_{\mathbf{P}}(w_{i+1}^{l,j})$ using local data.
  Device $j$ sends encrypted updates back to the gateway.
**end for**
Gateway computes partially-aggregated encrypted update: $\text{Enc}_{\mathbf{P}}(\sum_{j \in S_i^l} w_{i+1}^{l,j})$.
Gateway sends $\text{Enc}_{\mathbf{P}}(\sum_{j \in S_i^l} w_{i+1}^{l,j})$ to the server for full aggregation with other clusters.
Gateway receives encrypted fully-aggregated update $\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1})$ from the server.
Gateway partially decrypts encrypted fully-aggregated model: $\text{Dec}_{\mathbf{S}_l,\mathbf{P}}(\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1})$.
Gateway sends partially-decrypted fully-aggregated model weights back to the server for fully decryption.

---

**Gateway Side ($t$, $t \in \mathbb{T}$):**
**Require:** global average model weights $w_i$, TMHE threshold $\mathbf{T}_t$, FMHE global public key $\mathbf{P}$.
Gateway selects a subset of devices $S_i^t$ randomly.
**for all** selected devices $j \in S_i^t$ **in parallel do**
  Gateway sends global average model weights $w_i$ to device $j$.
**end for**
**for all** selected devices $j \in S_i^t$ **in parallel do**
  Device $j$ computes encrypted updates $\text{Enc}_{\mathbf{P}}(w_{i+1}^{t,j})$ using local data.
  Device $j$ sends encrypted updates back to the gateway.
**end for**
Gateway computes partially-aggregated encrypted update: $\text{Enc}_{\mathbf{P}}(\sum_{j \in S_i^t} w_{i+1}^{t,j})$
Gateway sends $\text{Enc}_{\mathbf{P}}(\sum_{j \in S_i^t} w_{i+1}^{t,j})$ to the server for full aggregation with other clusters.
Gateway receives encrypted fully-aggregated update $\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1})$ from the server.
Gateway selects $\mathbf{T}_t$ online devices in its cluster.
**for all** devices $j$ in cluster $t$ **in parallel do**
  Device $j$ partially decrypts encrypted fully-aggregated model $\text{Dec}_{sk_j^t,\mathbf{pk_t}}(\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1})$
  Device $j$ sends partially-decrypted updates back to the gateway.
**end for**
Gateway aggregates $\mathbf{T}_t$ partially-decrypted model and gets partially-decrypted fully-aggregated model: $\text{Dec}_{\mathbf{S}_t,\mathbf{P}}(\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1})$.
Gateway sends partially-decrypted fully-aggregated model weights back to the server for fully decryption.

**Algorithm 5** Training Protocol of H-MHE on H-FL (Device Layer)

**Device Side ($j$ in Cluster $l$, $l \in \mathbb{L}$):**
**Require:** global average model weights $w_i$, FMHE global public key $\mathbf{P}$, local data.
**for** $e = 1$ **to** $E$ **do**
  **for** batch $b \in$ local data **do**
    Compute gradient $\nabla L(w_i, b)$ on batch $b$
    Update weights: $w_i = w_i - \eta \nabla L(w_i, b)$
  **end for**
**end for**
Compute encrypted update: $\text{Enc}_{\mathbf{P}}(w_{i+1}^{l,j})$
Return encrypted update to the Gateway

---

**Device Side ($j$ in Cluster $t$, $t \in \mathbb{T}$):**
**Require:** global average model weights $w_i$, FMHE global public key $\mathbf{P}$, TMHE private key share $sk_j^t$, local data.
**for** $e = 1$ **to** $E$ **do**
  **for** batch $b \in$ local data **do**
    Compute gradient $\nabla L(w_i, b)$ on batch $b$
    Update weights: $w_i = w_i - \eta \nabla L(w_i, b)$
  **end for**
**end for**
Compute encrypted update: $\text{Enc}_{\mathbf{P}}(w_{i+1}^{t,j})$
Return encrypted update to the Gateway
Receive encrypted fully-aggregated model $\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1})$ from the gateway.
Partially decrypt the ciphertext: $\text{Dec}_{sk_j^t, \mathbf{pk_t}}(\text{Enc}_{\mathbf{P}}(|S_i| \cdot w_{i+1}))$.
Submit it to the gateway.

### 4.2.4 Security Analysis

In this section we will analyze the privacy guarantee of H-MHE against curious-but-honest adversaries. Collusion guarantee of FMHE is $n < N - 1$, meaning that in a two-tier FL setting which has $N$ clients, as long as there is less than $N - 1$ clients colluding, the data privacy of any parties is secure against Leaning-based Attacks. Naturally, we can derive that in our case as long as there is less than $K - 1$ clusters colluding, the data privacy of any client devices is secure against Leaning-based Attacks. Collusion on a cluster has two forms: 1) for clusters utilizing THE scheme ($t \in \mathbb{T}$), a cluster $t$ colluding means that at least $\mathbf{T}_t$ devices collude with each other to reveal the private key $\mathbf{S}_t$; 2) for clusters under Plain HE scheme ($l \in \mathbb{L}$), a cluster $l$ colluding means the third party key generator of this cluster shares its private key $\mathbf{S}_l$ with other colluders. If we assume that colluding with a gateway under Plain HE is easier than colluding with a THE cluster which has the highest threshold, then we can derive the exact collusion guarantee of H-MHE: $n < (\Sigma_{t \in \mathbb{T} \setminus \{\arg\max_i \mathbf{T}_i\}} \mathbf{T}_t) + |\mathbb{L}|$.

We can conclude that H-MHE provides high privacy guarantee, meanwhile mitigates the shortcommings of PMHE, FMHE, and TMHE in large-scale and unreliable networks. In other words, we 1) utilize TMHE and PMHE in the Device Layer to tolerate dropout and heterogeneous delay in IoT networks, and at the same time maintain a certain level of privacy guarantee by threshold; 2) we adapt FMHE in the Gateway Layer, which also leverage the fact that gateways are usually much more stable than IoT devices, so we can utilize the all-party-participation requirement of FMHE to provide high privacy guarantee, while avoiding the shortcoming that FMHE cannot tolerate dropout and delay.

### 4.2.5 Performance Analysis of Device Layer

In this section, we analyze the performance of devices in clusters running TMHE or PMHE, in terms of computation cost, communication cost, and storage cost.

#### Overhead during Setup

- **Computation cost** $O(N_t), t \in \mathbb{T}$ for clients under TMHE and $O(1)$ for those under PMHE. Each client's computation cost under TMHE can be broken up as: 1) generating secret shares, which takes $O(1)$ time; 2) performing secret sharing to generate private key shares and local public key, which takes $O(N_t)$ time.
- **Communication cost** $O(N_t^2), t \in \mathbb{T}$ for clients under TMHE and $O(1)$ for those under PMHE. For clients under TMHE, they have to send and receive secret shares to all other clients in the same cluster, so total cost is $O(N_t^2), t \in \mathbb{T}$.
- **Storage cost** $O(N_t), t \in \mathbb{T}$ for clients under THE and $O(1)$ for those under Plain HE.

**Overhead during Learning**

- **Computation cost** Given $m$ the size of the model, $O(m) + O_{training}, t \in \mathbb{T}$ is for clients under TMHE and $O(m) + O_{training}$ for those under Plain HE. Each client's computation cost under THE can be broken up as: 1) receiving and sending parameters and model weights, which take $O(1)$ time; 2) one decryption and one encryption, which take $O(2m)$ time. For those under PMHE, they only perform one encryption.
- **Communication cost** $O(m)$ for clients under TMHE and $O(m)$ for those under PMHE. For clients under THE, they receive plaintext model one time, send encrypted model two times, and receive encrypted model one time, so their communication cost is $O(4m+1)$. For PMHE, they receive plaintext model one time and send encrypted model one time, so total communication cost is $O(2m+1)$.
- **Storage cost** $O(m) + O_{training}$ for clients under THE and $O(m) + O_{training}$ for those under Plain HE.

### 4.2.6 Performance Analysis of Gateway Layer

In this section, we analyze the performance of gateways in clusters running TMHE or PMHE, in terms of computation cost, communication cost, and storage cost.

**Overhead during Setup**

- **Computation cost** $O(1)$ for gateways under both TMHE and PMHE.
- **Communication cost** $O(N_t), t \in \mathbb{T}$ for gateways under TMHE and $O(N_l), l \in \mathbb{T}$ for gateways under PMHE. The main communication is distributing public keys to client devices.
- **Storage cost** $O(1)$ for gateways under both TMHE and PMHE.

**Overhead during Learning**

- **Computation cost** $O(N_t \cdot m), t \in \mathbb{T}$ for gateways under TMHE and $O(N_l \cdot m), l \in \mathbb{L}$ for gateways under PMHE. For the sake of synchronization and faster convergence, we recommend employing Plain HE on large clusters while THE in small clusters.
- **Communication cost** $O(N_t \cdot m), t \in \mathbb{T}$ for gateways under THE and $O(N_l \cdot m), l \in \mathbb{L}$ for gateways under Plain HE.
- **Storage cost** $O(N_k \cdot m)$ for gateways under both TMHE and PMHE.

### 4.2.7 Performance Analysis of Server Layer

In this section, we analyze the performance of the central server in terms of computation cost, communication cost, and storage cost.

#### Overhead during Setup

- **Computation cost** $O(K)$, $K$ is the number of clusters.
- **Communication cost** $O(K)$.
- **Storage cost** $O(K)$.

#### Overhead during Learning

- **Computation cost** $O(K \cdot m)$.
- **Communication cost** $O(K \cdot m)$.
- **Storage cost** $O(K \cdot m)$.

## 4.3 Experiments

In this section, we evaluate the performance of the FL with H-MHE prototype system in comparison with current state-of-the-art multiparty homomorphic encryption schemes, and further test its tolerance against dropouts.

### 4.3.1 Heterogeneous Large-scale Network Simulation

We simulate a heterogeneous large-scale network for the testing of FL with H-MHE or other MHE protocols using `Golang`'s `Gorountines` and `Channels`. This is a distributed system with 100 devices and a central server for managing computation process, running FL to train a neural network on the benchmark image classification datasets: MNIST [36].

To test the resilience of H-MHE against dropouts, we simulate the heterogeneity of the network by dividing devices in the network to 4 categories. Each category $i$ has an *average dropout rate* $\mu_i$ and a *standard deviation of dropout rate* $\sigma_i$, and devices in the category sample their dropout rates from the normal distribution $\mathbb{N}(\mu_i, \sigma_i^2)$. For simplicity, we divide the network into four equal parts. The heterogeneity setting of the network is shown in Table 4.1.

| Category # | $\mu$ | $\sigma$ | Number of devices | Threshold of H-MHE clusters |
|---|---|---|---|---|
| 1 | 0.5 | 0.1 | 25 | 4 |
| 2 | 0.2 | 0.1 | 25 | 9 |
| 3 | 0.05 | 0.1 | 25 | 17 |
| 4 | 0.01 | 0.1 | 25 | 20 |

Table 4.1: Heterogeneity setting of the network simulation. We divide the network to four categories, each contains 25 devices. For devices in each category, their dropout rate are sampled from the normal distribution $\mathbb{N}(\mu, \sigma^2)$. The last column is the thresholds of H-HME clusters according to the cluster network setting.

We will compare H-MHE to FMHE, TMHE, and PMHE on FL tasks in terms of model accuracy, time overhead, communication overhead, and resilience to dropouts of devices. For TMHE, we set the threshold half of the total number of devices. For H-MHE, we divide the network to 4 clusters according to the categorization, and the sum of thresholds of all clusters equals to the threshold of TMHE for fairness of comparison. We set the cluster thresholds for each cluster in H-MHE according to the cluster network setting, i.e., average dropout rate, for better resilience, shown as the last column in Table 4.1. All experiments are run with five seeds from 1 to 5 for reproducibility.

### 4.3.2 Federated Learning Setting

We present the architecture of the neural network models and additional details on the experimental setup in Table 4.2. All our experiments were conducted on the following hardware: 1 11th Gen Intel Core i9-11900H @ 2.50GHz and RAM 32.0GB. We use a compact notation to define the models as in [20].

### 4.3.3 Experimental Results

The experimental results on the 100-device network simulation are shown in Table 4.3.

**Resilience to dropouts.** From experimental results we notice that with an approperiate arrangement for cluster thresholds in H-MHE, H-MHE can achieve asymptotic dropout resilience as TMHE, PMHE, and even raw FL. In our results, the average failure rate of H-MHE is 0.000625, which is negligible during federated learning. On the contrast, FMHE provides no resilience against dropouts, so all of its training round failed.

**No accuracy drop.** In our experimental results we only detect negligible accuracy drop in H-MHE in comparison with raw FL, which can be the cause of randomness.

| *Dataset* | **MNIST** |
|---|---|
| *Architecture* | (1,28,28)-L(50)-R-L(10) |
| *Loss* | Cross Entropy |
| *Preprocessing* | Normalization |
| *Aggregation rules* | Average |
| *Number of steps* | $T = 320$ |
| *Batch size* | $b = 32$ |
| *Learning rate* | $\gamma_t = 0.05$ |
| *Adam parameter* | $\beta = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ |
| *Random seeds* | $1, 2, 3, 4, 5$ |
| *Encryption scheme* | RNS-CKKS or None |
| *Encryption parameter* | $log_2(N) = 14, log_2(QP) = 390, L = 9$ or None |
| *Security level* | 128 Quantum or None |

Table 4.2: Model Architectures, hyperparameters, and distributed settings for MNIST experiments. L(#*outputs*) for a fully-connected linear layer with #*outputs* dimensions of output, R for ReLU activation.

| Scheme | Accuracy | Setup Time (s) | Training Time (s) | Number of Failure Rounds |
|---|---|---|---|---|
| None | 0.930 | 0 | 189.0 | 0 |
| PMHE | 0.926 | 0.0559 | 564.5 | 0 |
| FMHE | 0.110 | 1.52 | 628.8 | 320 |
| TMHE | 0.926 | 70.18 | 1043.2 | 0 |
| H-MHE | 0.922 | 5.73 | 916.3 | 0.2 |

Table 4.3: Experimental results about best accuracy, time overhead during cryptosystem setup and training, and the number of rounds failing to decrypt. Each experiment runs 5 times and the average is shown in the table.

**Significant speedup during setup and training process.** We detect 12.25× speedup during cryptosystem setup phase and 1.14× speedup during training process in comparison with TMHE, which is a big advancement.

**Scalability.** We also notice in our experiment that the time overhead and communication cost are only related to the size of the biggest cluster in H-MHE system. So as long as we constrain the size of clusters, and only add more clusters but not enlarge existing clusters when there are more devices, H-MHE can always maintain good efficiency.

From experimental results we can draw these conclusions:

- Compared to TMHE, H-MHE distributes the threshold T to different clusters, meaning, $T =$

$\Sigma T_i$ ($T_i$: the threshold of cluster i. The threshold of FMHE can be seen as Ni, the total number of parties in cluster i. The threshold of PMHE can be seen as 1), which provides flexibility for different network environments and computation resources and so a more flexible level of resilience to delays or dropouts, while preserving the same guarantee against collusion. For example, clusters with weaker network reliability and less computation resources can choose small threshold, while clusters with reliable network and sufficient computation power can choose much bigger threshold, because the bigger is the threshold, the more required are reliable network and higher computation power. Thus, this feature makes H-MHE much more suitable for large-scale and heterogeneous networks than any existing MHE schemes.

- Splitting the networks to several clusters significantly reduces the computation and communication cost, for a cluster isolates a set of parties from all other parties in the rest of the networks, so that the computations and communications that should happen with parties in the whole networks now only happen with parties within a single cluster. This feature can be observed during any operations needing disclosure or broadcast of information, e.g., procedure of thresholdization.

- Clustering also provides isolation among different parties, meaning, the configuration changing (adding, changing, or removing a party) of a cluster will only have very little influence on other clusters (updating the public key). The re-setup only happens within the cluster with configuration changes. On the contrary, the whole system must be re-setup when there is a configuration change in the existing MHE systems.

# Chapter 5

# Conclusion and Future Work

Traditional HE schemes, designed for cross-silo FL, falter in addressing IoT-specific challenges: 1) large scale; 2) heterogeneity; 3) unreliability; 4) low overhead. To bridge this gap, we propose a pioneering Hierarchical Multiparty Homomorphic Encryption protocol tailored for Hierarchical FL. Our approach innovates by extending Threshold Multiparty Homomorphic Encryption to a hierarchical and hybrid access structure, aiming to establish a precedent for effective and practical privacy-preserving FL in IoT environments.

We encompassed a comprehensive investigation, implementation, and testing of the H-MHE protocol, marking a significant stride toward securing FL in IoT applications. By addressing the nuanced requirements of IoT scenarios, our work aspires to fortify the privacy landscape of FL, ensuring robust data protection while maintaining model performance. In our experiments, we simulated a 100-device network and tested H-MHE with 4 clusters in the network, with MNIST dataset as the FL task. H-MHE achieved a 12.25× speedup during cryptosystem setup phase and 1.14× speedup during training phase compared to the state-of-the-art TMHE framework, while maintaining asymptotic resilience against network unreliability. Notably, the trained model maintains its performance in comparison to models trained by conventional raw FL framework without encryption. The evaluation demonstrates that the scheme outperforms other innovations in resilience, and communication and computational cost, while preserving model accuracy in cross-device FL scenarios.

However, although the Hierarchical Multiparty Homomorphic Encryption (H-MHE) scheme proposed in this paper has successfully achieved the intended research objectives, there are still areas where our work can be improved.

First, integrating Differential Privacy (DP) techniques with the Hierarchical Multiparty Homomorphic Encryption (H-MHE) protocol could provide comprehensive privacy protection for both the learning process and the aggregated model. While the current H-MHE framework effectively

secures model updates during the Federated Learning (FL) process, the aggregated model remains exposed to potential privacy threats. By embedding DP into the encryption process, it may be possible to obscure individual data contributions within the aggregated model, further mitigating the risk of privacy breaches. Future work could explore the trade-offs between privacy guarantees and model accuracy when combining H-MHE with DP, as well as the impact on computational and communication efficiency.

Secondly, despite the significant reductions in time and communication costs achieved by H-MHE compared to the state-of-the-art TMHE and FMHE frameworks, the overall cost remains prohibitively high for real-world deployment in IoT environments. This highlights a fundamental challenge with Homomorphic Encryption (HE) technologies, where even the most lightweight solutions (e.g., PMHE) still incur substantial computational overhead. To address this, further research is needed to optimize the H-MHE protocol, perhaps by leveraging advances in hardware acceleration, such as GPU or FPGA implementations, or exploring more efficient cryptographic primitives that maintain the security guarantees of HE while reducing computational demands.

Additionally, while the current project has successfully implemented a prototype system based on H-MHE for Hierarchical Federated Learning (HFL), this system is not yet ready for direct deployment in actual IoT networks. Future work should focus on the software engineering aspects required to transition from a research prototype to a production-ready system. This may involve developing robust, scalable, and user-friendly software tools and libraries, as well as conducting extensive testing in diverse and realistic IoT environments. Ensuring that the system can handle the variability and complexity of real-world IoT deployments will be crucial for its practical adoption.

Moreover, although the experimental evaluation of H-MHE was thorough, it was conducted in a simulated network environment using `Goroutines` and `channels` to mimic network communication. To obtain more rigorous, practical, and scientifically sound results, future studies should implement and test the proposed H-MHE framework in real IoT scenarios. This could involve deploying the system in a range of IoT environments, from smart homes to industrial IoT applications, to assess its performance, scalability, and security under realistic conditions. Such testing would also help identify any unforeseen challenges or limitations that may arise in real-world deployments, providing valuable insights for further refinement of the protocol.

In conclusion, while the proposed H-MHE protocol represents a significant advancement in securing Federated Learning in IoT environments, it also opens up new avenues for research and development. By addressing the remaining challenges and building upon the foundation laid in this work, future research can further enhance the practicality, security, and efficiency of privacy-preserving FL in IoT networks.

# Bibliography


[1] 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*. 2016, pp. 308–318.

[2] Hanane Allioui and Youssef Mourdi. "Exploring the Full Potentials of IoT for Better Financial Growth and Stability: A Comprehensive Survey". In: *Sensors* 23.19 (2023). ISSN: 1424-8220. DOI: 10.3390/s23198015. URL: https://www.mdpi.com/1424-8220/23/19/8015.

[3] Youssef Allouah, Rachid Guerraoui, Nirupam Gupta, Rafaël Pinot, and John Stephan. "On the Privacy-Robustness-Utility Trilemma in Distributed Learning". In: *International Conference on Machine Learning*. 202. 2023.

[4] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. "Multiparty computation with low communication, computation and interaction via threshold FHE". In: *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*. Springer. 2012, pp. 483–501.

[5] Dimitri Bertsekas and John Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. "Towards federated learning at scale: System design". In: *Proceedings of machine learning and systems* 1 (2019), pp. 374–388.

[7] 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*. 2017, pp. 1175–1191.

[8] Jean-Philippe Bossuat, Rosario Cammarota, Jung Hee Cheon, Ilaria Chillotti, Benjamin R. Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, Xianhui Lu, Carsten Maple, Alberto Pedrouzo-Ulloa, Rachel Player, Luis Antonio Ruiz Lopez, Yongsoo Song, Donggeon Yhee, and Bahattin Yildiz. *Security Guidelines for Implementing*

*Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2024/463. https://eprint.iacr.org/2024/463. 2024. URL: https://eprint.iacr.org/2024/463.

[9] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. "Extracting training data from large language models". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 2633–2650.

[10] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. "Efficient Multi-Key Homomorphic Encryption with Packed Ciphertexts with Application to Oblivious Neural Network Inference". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 395–412. ISBN: 9781450367479. DOI: 10.1145/3319535.3363207. URL: https://doi.org/10.1145/3319535.3363207.

[11] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. "Secureboost: A lossless federated learning framework". In: *IEEE Intelligent Systems* 36.6 (2021), pp. 87–98.

[12] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 409–437. ISBN: 978-3-319-70694-8.

[13] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic encryption for arithmetic of approximate numbers". In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer. 2017, pp. 409–437.

[14] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. "Differential privacy-enabled federated learning for sensitive health data". In: *arXiv preprint arXiv:1910.02578* (2019).

[15] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems* 25 (2012).

[16] Yvo G Desmedt. "Threshold cryptography". In: *European Transactions on Telecommunications* 5.4 (1994), pp. 449–458.

[17] Alan A.A. Donovan and Brian W. Kernighan. *The Go Programming Language*. 1st. Addison-Wesley Professional, 2015. ISBN: 0134190440.

[18] Weidong Du, Min Li, Liqiang Wu, Yiliang Han, Tanping Zhou, and Xiaoyuan Yang. "A efficient and robust privacy-preserving framework for cross-device federated learning". In: *Complex & Intelligent Systems* 9.5 (2023), pp. 4923–4937.

[19] Patrik Ehrencrona, Melle Koning, Yonder Blue, Moritz Spindelhirn, and Vladislav A. Proskurov. *go-deep*. Online: https://github.com/patrikeh/go-deep. Aug. 2024.

[20] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Louis Alexandre Rouault. “Distributed momentum for byzantine-resilient stochastic gradient descent”. In: *9th International Conference on Learning Representations (ICLR)*. 2021.

[21] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, and Shaza Zeitouni. “SAFELearn: Secure Aggregation for private FEderated Learning”. In: *2021 IEEE Security and Privacy Workshops (SPW)*. 2021, pp. 56–62. DOI: 10.1109/SPW53761.2021.00017.

[22] 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*. 2015, pp. 1322–1333.

[23] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. “Internet of things: A survey on enabling technologies, protocols, and applications”. In: *IEEE communications surveys & tutorials* 17.4 (2015), pp. 2347–2376.

[24] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In: *Symposium on the Theory of Computing*. 2009. URL: https://api.semanticscholar.org/CorpusID:947660.

[25] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. ISBN: 9781605585062. DOI: 10.1145/1536414.1536440. URL: https://doi.org/10.1145/1536414.1536440.

[26] Neveen Mohammad Hijazi, Moayad Aloqaily, Mohsen Guizani, Bassem Ouni, and Fakhri Karray. “Secure Federated Learning With Fully Homomorphic Encryption for IoT Communications”. In: *IEEE Internet of Things Journal* 11.3 (2024), pp. 4289–4300. DOI: 10.1109/JIOT.2023.3302065.

[27] 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*. 2017, pp. 603–618.

[28] Hsiang Hsu, Natalia Lucienne Martinezgil, Martin Bertran, Guillermo Sapiro, and Flavio Calmon. “A Survey on Privacy from Statistical, Information and Estimation-Theoretic Views”. In: *IEEE BITS the Information Theory Magazine*. 2021. URL: https://ieeexplore.ieee.org/document/9524532.

[29] Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, and Yanmin Gong. “Personalized federated learning with differential privacy”. In: *IEEE Internet of Things Journal* 7.10 (2020), pp. 9530–9539.

[30] Weizhao Jin, Yuhang Yao, Shanshan Han, Carlee Joe-Wong, Srivatsan Ravi, Salman Avestimehr, and Chaoyang He. “FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System”. In: *arXiv preprint arXiv:2303.10837* (2023).

[31] M. I. Jordan and T. M. Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245 (2015), pp. 255–260. DOI: 10.1126/science.aaa8415. eprint: https://www.science.org/doi/pdf/10.1126/science.aaa8415. URL: https://www.science.org/doi/abs/10.1126/science.aaa8415.

[32] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. “Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning”. In: *arXiv preprint arXiv:2009.11248* (2020).

[33] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. “Advances and open problems in federated learning”. In: *Foundations and trends® in machine learning* 14.1–2 (2021), pp. 1–210.

[34] *Lattigo v5*. Online: https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA. Nov. 2023.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[36] Yann LeCun and Corinna Cortes. “The mnist database of handwritten digits”. In: 2005.

[37] Hang Li. “Introduction to Machine Learning and Supervised Learning”. In: *Machine Learning Methods*. Springer, 2023, pp. 1–37.

[38] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. “Client-edge-cloud hierarchical federated learning”. In: *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE. 2020, pp. 1–6.

[39] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B Letaief. “Hierarchical federated learning with quantization: Convergence analysis and system design”. In: *IEEE Transactions on Wireless Communications* 22.1 (2022), pp. 2–18.

[40] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “On ideal lattices and learning with errors over rings”. In: *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*. Springer. 2010, pp. 1–23.

[41] Jing Ma, Si-Ahmed Naas, Stephan Sigg, and Xixiang Lyu. “Privacy-preserving federated learning based on multi-key homomorphic encryption”. In: *International Journal of Intelligent Systems* 37.9 (2022), pp. 5880–5901.

[42] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[43] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. “Exploiting unintended feature leakage in collaborative learning”. In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, pp. 691–706.

[44] Payman Mohassel and Peter Rindal. “ABY3: A mixed protocol framework for machine learning”. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 35–52.

[45] Payman Mohassel and Yupeng Zhang. “Secureml: A system for scalable privacy-preserving machine learning”. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 19–38.

[46] Arsalan Mosenia and Niraj K Jha. “A comprehensive study of security of internet-of-things”. In: *IEEE Transactions on emerging topics in computing* 5.4 (2016), pp. 586–602.

[47] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. “An Efficient Threshold Access-Structure for RLWE-Based Multiparty Homomorphic Encryption”. In: *Journal of Cryptology* 36.2 (2023), p. 10. DOI: 10.1007/s00145-023-09452-8. URL: https://doi.org/10.1007/s00145-023-09452-8.

[48] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. “An efficient threshold access-structure for rlwe-based multiparty homomorphic encryption”. In: *Journal of Cryptology* 36.2 (2023), p. 10.

[49] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. “Multiparty homomorphic encryption from ring-learning-with-errors”. In: *Proceedings on Privacy Enhancing Technologies* 2021.4 (2021), pp. 291–311.

[50] Maxence Noble, Aurélien Bellet, and Aymeric Dieuleveut. “Differentially private federated learning on heterogeneous data”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 10110–10145.

[51] Oracle. *What Is the Internet of Things?* https://www.oracle.com/internet-of-things/what-is-iot/. Accessed: 2024-08-16.

[52] Pascal Paillier. “Public-key cryptosystems based on composite degree residuosity classes”. In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, pp. 223–238.

[53] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. “Privacy-Preserving Deep Learning via Additively Homomorphic Encryption”. In: *IEEE Transactions on Information Forensics and Security* 13.5 (2018), pp. 1333–1345. DOI: 10.1109/TIFS.2017.2787987.

[54] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. “Privacy-preserving deep learning: Revisited and enhanced”. In: *Applications and Techniques in Information Security: 8th International Conference, ATIS 2017, Auckland, New Zealand, July 6–7, 2017, Proceedings*. Springer. 2017, pp. 100–110.

[55] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. “The future of digital health with federated learning”. In: *NPJ digital medicine* 3.1 (2020), pp. 1–7.

[56] David Ruppert. *The elements of statistical learning: data mining, inference, and prediction*. 2004.

[57] Adi Shamir. "How to share a secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[58] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. "Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data". In: *Scientific reports* 10.1 (2020), p. 12598.

[59] Reza Shokri and Vitaly Shmatikov. "Privacy-preserving deep learning". In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1310–1321.

[60] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. "Membership inference attacks against machine learning models". In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18.

[61] Jinhyun So, Ramy E Ali, Başak Güler, Jiantao Jiao, and A Salman Avestimehr. "Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 8. 2023, pp. 9864–9873.

[62] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[63] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. "Stealing machine learning models via prediction {APIs}". In: *25th USENIX security symposium (USENIX Security 16)*. 2016, pp. 601–618.

[64] 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*. 2019, pp. 1–11.

[65] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. "Beyond inferring class representatives: User-level privacy leakage from federated learning". In: *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE. 2019, pp. 2512–2520.

[66] Febrianti Wibawa, Ferhat Ozgur Catak, Murat Kuzlu, Salih Sarp, and Umit Cali. "Homomorphic encryption and federated learning based privacy-preserving cnn training: Covid-19 detection use-case". In: *Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference*. 2022, pp. 85–90.

[67] Runhua Xu, Nathalie Baracaldo, and James Joshi. "Privacy-preserving machine learning: Methods, challenges and directions". In: *arXiv preprint arXiv:2108.04417* (2021).

[68] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. "Federated machine learning: Concept and applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.

[69] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. “Privacy risk in machine learning: Analyzing the connection to overfitting”. In: *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE. 2018, pp. 268–282.

[70] Xiaofan Yu, Lucy Cherkasova, Harsh Vardhan, Quanling Zhao, Emily Ekaireb, Xiyuan Zhang, Arya Mazumdar, and Tajana Rosing. “Async-HFL: Efficient and Robust Asynchronous Federated Learning in Hierarchical IoT Networks”. In: *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*. IoTDI ’23. San Antonio, TX, USA: Association for Computing Machinery, 2023, pp. 236–248. ISBN: 9798400700378. DOI: 10.1145/3576842.3582377. URL: https://doi.org/10.1145/3576842.3582377.

[71] 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)*. 2020, pp. 493–506.

[72] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. “idlg: Improved deep leakage from gradients”. In: *arXiv preprint arXiv:2001.02610* (2020).

[73] Junyi Zhu and Matthew B Blaschko. “R-GAP: Recursive Gradient Attack on Privacy”. In: *International Conference on Learning Representations*. 2020.

[74] Ligeng Zhu, Zhijian Liu, and Song Han. “Deep leakage from gradients”. In: *Advances in neural information processing systems* 32 (2019).