October 2022

Improving Wireless Networking from the Learning and Security Perspectives

Zhe Qu
University of South Florida

Improving Wireless Networking from the Learning and Security Perspectives

by

Zhe Qu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Zhuo Lu, Ph.D.
Nasir Ghani, Ph.D.
Ismail Uysal, Ph.D.
Attila A. Yavuz, Ph.D.
Jie Xu, Ph.D.

Date of Approval:
October 11, 2022

Keywords: Federated Learning, Edge Computing, Client-Selection, Poisoning Data Attack, Key Establishment

Dedication

This dissertation is dedicated to my brilliant and magnificent wife, Ying Chen, and my parents, Fucheng Qu and Jing Yu, who give me unreserved support throughout this journey.

Acknowledgments

As we all know, a Ph.D. career is a quite challenging and struggling journey, but for me, I also obtained many favorable experiences. First of all, I would like to especially thank my major advisor Dr. Zhuo Lu, who gives me insightful guidance and significant support throughout my whole Ph.D. career. Without his diligent help both in normal life and research, I could not successfully finish my Ph.D. degree. His patience, his motivation, and his down-to-earth working style, as well as his extensive knowledge, influenced me a lot. Definitely, I am very proud and appreciative that he can be my advisor.

Xingyu Li, my academic brother, during these years, provided me with much knowledge from the other side and shared with me many insightful understandings of research. We took many enjoyable conversations on research and life experiences. And I hope him to seek a wonderful job and enjoy his rest Ph.D. period.

Specifically, I appreciate all my committee members, Dr. Ismail Uysal, Dr. Nasir Ghani, Dr. Attila A. Yavuz, Dr. Jie Xu. I feel so glad and honored to have them as my committee members and for their efforts and suggestions.

I also give my sincere thank to my friends and my collaborators in my Ph.D. career. They are Dr. Yao Liu, Dr. Bo Tang, Dr. Shangqing Zhao, Dr. Zhengping Luo, Dr. Tao Hou, Dr. Mohammed Alrowaily, Rui Duan, Xiao Han, Junjie Xiong, Xiaowen Li, Yuwen Cui, Wenwei Zhao, Jiahao Xue, Sen Wang, Keyu Chen, Mingchen Li, Jingwei He, etc.. To this end, I am very valuable with my three lovely kitties: Yuhuan, Kassie, and Yertle, they stayed with me all my happy and struggling Ph.D. lifetime.

Table of Contents

List of Tables

List of Figures

Abstract

Due to the high development of wireless networking and artificial intelligence, most of the data are generated from mobile devices, which distribute in different environments. As such, how to improve the performance of machine learning-based networking and its security should be carefully considered. To reduce the communication burden and protect private information from users, Federated Learning (FL) is a possible solution for learning-based wireless networking. Although FL achieves much success until now, it also remains some specific issues to be solved. In this dissertation, we propose two FL wireless networking frameworks and discuss two potential security issues.

In the FL wireless network, due to the single server architecture, the server processes the global computing after receiving all local model updates, which means the training period high depends on the slowest clients. Because we cannot avoid all clients obtaining qualified wireless channels, it is necessary to design new FL architectures to improve the training performance from a networking perspective. Therefore, we design a new multi-server FL architecture and propose an FL algorithm for this architecture called MS-FedAvg. Next, by leveraging contextual information, we propose an online selection policy called COCS, which is based on the contextual combinatorial multi-armed bandits.

For the network security design, we first investigate the targeted model poisoning attack. To cope with this security issue, we design a new defense strategy, called LoMar, in FL based on kernel function, which can distinguish the distribution difference between honest and malicious clients. The common assumption of wireless key generation considers that the wireless channel information is sufficiently random. By leveraging the not random property,

we design an efficient attack, named MLTS. Based on MLTS, we propose a design guideline for how to use the wireless channel to generate the secret key.

Chapter 1: Introduction

Thanks to advanced mobile technologies, e.g., the Internet of Things (IoT), cellular networks, 5G, and beyond, machine learning based wireless networking has been widely developed. Federated Learning (FL) is one of the most popular learning-based networking frameworks, which can be efficiently leveraged on the edge computing system. Different from the conventional machine learning technique, FL communicates learning model updates instead of training data. The advantage of FL can be divided into two aspects: communication burden reduction and privacy-preserving. Although many studies design new FL algorithms to improve the performance from different perspectives, there are also many remaining problems for FL, especially in FL network architecture and security. In this dissertation, we develop four mechanisms for improving FL performance from networking and security perspectives.

1.1 Learning Based Wireless Networking

Although FL has been demonstrated to achieve high success and efficiency for distributed machine learning, one of the most important problems is how to reduce the communication latency. Since the FL server requires to receive all local model updates and then process the aggregation, the communication period depends on the slowest client. To address this issue, we propose the multi-server FL architecture, which can significantly reduce the distance between clients and regional servers. Another strategy to reduce the communication latency is to design a client selection policy, and hence we develop an online client selection policy for hierarchical FL based on contextual combinatorial multi-armed bandits.

1.1.1 Multi-Server FL with Overlapping Area

With the explosive growth in the number of mobile phones and Internet of Things (IoT) devices, a tremendous amount of data today is being generated at the network edge in a distributed manner. Sending this data to the cloud for processing not only puts a huge burden on the network but also raises serious data privacy concerns. Federated Learning (FL) [66, 107, 146] recently emerged as a distributed Machine Learning (ML) architecture that keeps all the training data on individual clients, thereby protecting client data privacy and mitigating network congestion.

In its most common form, FL is an iterative process wherein each communication round, clients train local ML models using their local training datasets based on the current global ML model, and then the server aggregates the local models uploaded by the clients to update the global model. Because FL is trained on distributed datasets and often involves many communication rounds of model data exchange between the clients and the server, improving the communication efficiency between clients and central server [196, 49] and handling heterogeneous local dataset distribution in each client [80, 181] are two biggest challenges of FL and have received a large amount of research attention.

Although promising progress has been made, existing FL architectures and algorithms dominantly focus on the single-server system. Most FL studies consider that clients should download and upload the learning models with the central server repeatedly in each communication round. This communication strategy may suffer a large communication delay in large-scale FL systems where many clients may be far away from the server [101, 112]. This large delay between the server and the clients directly prolongs the learning time of the existing single-server-based FL system, especially when the server is placed on the cloud. As increasingly many applications are delay-sensitive, e.g., autonomous driving and wearable health monitoring, new FL architectures that involve multiple servers have been proposed to reduce the communication latency between the server and the clients.

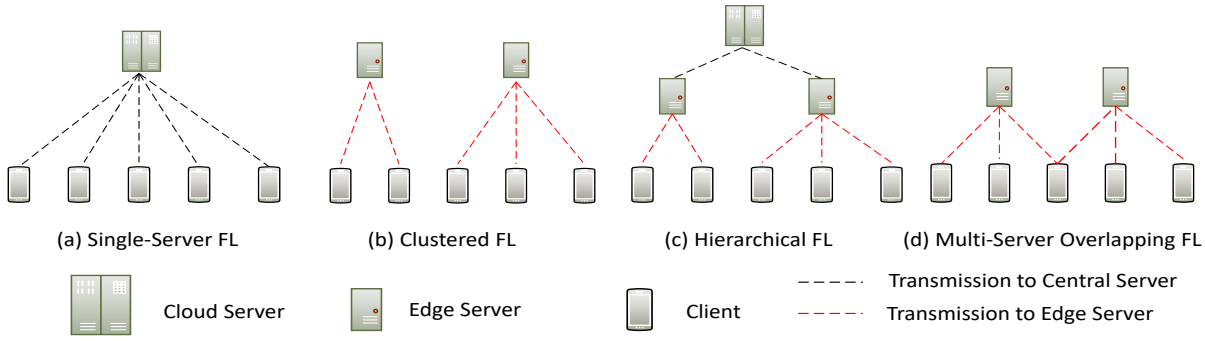


Figure 1.1: Description of FL network architectures.

To reduce the communication latency of FL, there are two main multiple servers FL approaches: (1) Hierarchical FL (HFL) [86, 92, 159] introduces a hierarchical structure for model training where multiple edge servers are used to collect and aggregate local model updates from clients in their respective service areas and then send the aggregated result to the cloud server for final aggregation. However, since the model exchange between the edge servers and the cloud server is still required, HFL can still result in a long training delay when the propagation latency between the edge servers and the cloud server is large. (2) Clustered FL (CFL) [174, 74, 38] divides clients into different clusters, and trains a separate ML model for each cluster. However, a re-clustering computation may be required in many communication rounds, thereby significantly increasing the training complexity and time. In addition, some existing studies ignore the physical network connectivity constraints – a client may connect to only a subset of servers.

In [50], a new FL architecture utilizing multiple servers is studied, which exploits the realistic deployment of 5G-and-beyond networks where a client can be located in the overlapping coverage areas of multiple servers. The network architectures of single-server FL, HFL, clustered FL and our proposed multi-server FL are shown in Fig. 1. The key idea is that clients download multiple models from all the edge servers they can access and train their local models based on the average of these models. Such architecture has two main advantages. First, by performing model averaging on the client side, each server indirectly

accesses the trained local models of clients not in its coverage while incurring a small model upload and download delay. Specifically, the broadcasting technique will not increase the communication burden. Second, instead of training multiple local models based on multiple downloaded models, each client only trains a single local model based on the average of the downloaded models, thereby avoiding extra computation and communication costs. Since the clients in overlapping areas should tackle multiple training models at the same time, the extra computation only comes from the averaging calculation, which is small compared to the local training process, and it can be negligible. Although [50] developed an algorithm for this architecture and empirically validated its effectiveness, they only proposed the strongly-convex loss function, which is very restricted, since most learning models are non-convex, e.g., neural network. In addition, the convergence results cannot show the impact on overlapping areas. In this dissertation, we improve upon this architecture, propose a new algorithm with two-sided learning rates, and provide a theoretical convergence analysis of the more general non-convex loss function. In summary, we highlight our main contributions as follows: 1) We develop a novel MS-FedAvg algorithm on this multi-server FL architecture, based on the two-sided learning rates FedAvg. 2) We study the convergence in the coverage area, which we call the region, of each server. For non-convex loss functions and non-iid datasets, we provide convergence analysis for full and unbiased partial client participation strategies, respectively. Our results are better than the existing multi-server FL algorithms and also reveal how the overlapping coverage affects the convergence in each region. 3) To further improve the convergence speed of MS-FedAvg, we develop a biased partial client participation strategy where clients may not be selected proportionally to the number of clients in different coverage areas. Our analysis shows that the degree of bias results in a trade-off between convergence rate and accuracy. 4) We conduct extensive experiments on multiple datasets under different multi-server FL network architectures and hyper-parameters. The experimental results show that our MS-FedAvg algorithm outperforms the compared benchmarks from accuracy and convergence perspectives.

1.1.2 Context-Aware Online Client Selection for HFL

Federated Learning (FL) [107, 80, 62] has become an attractive ML framework to address the growing concerns of transmitting private data from distributed clients (e.g., mobile devices) to a central cloud server by leveraging the ever-increasing storage and computing capabilities of the client devices. In each FL round, clients train local models using their local data and the cloud server aggregates local model updates to form a global model. Because only local model information is exchanged in FL rather than the local data, FL preserves the data privacy of the clients and hence has found applications in a wide range of problems, such as next-word prediction [199] and image classification [47].

The main bottleneck that limits the performance of FL is the delay variability among individual clients due to their local training and model data transfer via the wireless network. In standard FL, the cloud server has to wait until receiving the training updates from all the clients before processing any next step. Therefore, straggler clients who have unfavorable wireless links or low computation capabilities may dramatically slow down the whole FL process [101, 182]. This is the so-called "straggler effect". Various approaches have been proposed to mitigate the "straggler effect". For example, model quantization [133, 142] and gradient sparsification [147] schemes aim to directly reduce the transferred data size and the model training complexity, thereby reducing all clients' training and transmission delay. Asynchronous FL [168, 82] allows clients to train and asynchronously upload training data, and hence the cloud server does not have to wait for the slow clients to process the next step. Another mainstream and proven effective approach to address the straggler problem is client selection, which reduces the probability of straggler clients participating in FL by judiciously selecting clients in every FL round. However, these mechanisms mainly focus on the traditional FL and mitigate the straggler effect based on designing new mechanisms, which cannot solve the straggler effect from FL wireless networks (e.g., the large distance of clients-Cloud Server (CS) and unstable connection). Thanks to the hierarchical architecture, some existing studies [92, 93, 159] propose Hierarchical FL (HFL) including multiple Edge

Servers (ESs) which reside between the single CS and a large number of clients. Instead of communicating with the CS, clients in HFL only need to download/upload the training model updates to the nearest ESs, which significantly reduces the communication time of the slowest client located far from the CS and provides a more stable connection to save training time, which has been demonstrated to achieve a faster convergence speed than FL architecture both theoretically [92, 159] and empirically [93].

Although several learning algorithms have been designed for HFL [92, 93, 159], simplifying assumptions have been made that all clients participate in each round of model parameter aggregation. This is weak for the straggler effect, and hence it is necessary to design a new client selection mechanism for HFL. However, it is not straightforward to apply existing client selection solutions [158, 176, 182] to HFL due to several unique challenges that HFL faces. Firstly, since the service area of an ES is much more restricted than CS and contains overlapping areas, the accessible clients of each ES are time-varying. This time-varying characteristic makes the client behavior of opportunistic communication more complicated, and Network Operator (NO) must carefully select the client to the corresponding ES in the overlapping area. Secondly, since the advantage of HFL is to deal with the straggler problem, how to design an efficient client selection policy is more important than traditional FL. Thirdly, the client selection decision needs to be determined based on many uncertainties in the HFL network conditions (e.g., the traffic pattern of client-ES pair and available computation resources of clients), which affect training performance in previously unknown ways. Therefore, a learning-based client selection policy is preferred to a solely optimization-based policy.

In this dissertation, we investigate the client selection problem for HFL and propose a new learning-based policy, called Context-aware Online Client Selection (COCS). COCS is developed based on a novel Multi-Armed Bandit (MAB) framework called Contextual Combinatorial MAB (CC-MAB) [29, 28]. COCS is contextual because it allows clients to use their computational information (e.g., available computation resources), and the client-

ES pairs transmission information (e.g., bandwidth and distance). COCS is combinatorial because NO selects a subset of client-ES pairs and attempts to maximize the training utilities (i.e., select as many as clients in each round) by optimizing the client selection decision. To the best of our knowledge, COCS policy is the first client selection decision for HFL. In summary, we highlight the contributions as follows: 1) We formulate a client selection problem for HFL, where NO needs to select clients for ESs clients to process the local training to make more clients be received by ESs before the deadline under a limited budget. To improve the convergence speed for HFL, the client selection decision has a three-fold problem: (i) estimate the local model updates successfully received by ESs with cold-starts, (ii) decide whether a client should be selected to a certain ES due to time-varying connection conditions, and (iii) optimize how to pay computation resources on clients to maximize the utility under limited budgets. 2) Due to the a priori uncertain knowledge of participated clients, the client selection problem is formulated as a CC-MAB problem. An online learning policy called Context-aware Online Client Selection (COCS) is developed, which leverages contextual information such as downloading channel state and local computing time over aggregation round for making a decision. For the strongly convex HFL, we analyze the utility loss of COCS, termed regret, compared to the Oracle solution that knows the exacted information of participated clients. A sublinear regret bound is derived for the proposed COCS policy, which implies that COCS can produce asymptotically optimal client selection decisions for HFL. 3) For non-convex HFL, the utility function of the convergence speed is quadratically related to the number of participated clients. By assuming that the information of each client-ES pair is perfectly known by NO, we show that the client selection problem is a submodular maximization problem with M knapsack and one matroid constraint, where M is the number of ESs. We use the Fast Lazy Greedy (FLGreedy) algorithm [10] to approximate the optimal solution with a performance guarantee. To this end, the analysis shows that the COCS policy also achieves a sublinear regret.

1.2 Wireless Networking Security

The security issue of wireless networking is another important part to be considered. For learning-based wireless networking, because we cannot guarantee that all clients are honest, attackers may control some clients and achieve some attack goals. Specifically, one of the most popular attacks in FL is targeted model attacks, which aim to degrade the learning performance of part labels. In addition, this attack model is not easy to be detected, and hence it is necessary to design a new defense strategy. Key generation from a wireless channel is a common strategy due to channel randomness. However, it is well known that when the two consecutive signals are within the coherent time, the channel state information is not sufficiently random. As a result, attackers may leverage this property to successfully guess the secret key with a high probability and obtain a serious security issue. In contrast, existing studies do not well explore how to generate a secret key from a wireless channel. Therefore, we should carefully design a guideline to determine how to use the wireless channel information for key generation from both efficiency and security perspectives.

1.2.1 LoMar: A Local Defense Against Poisoning Attack on FL

Federated Learning (FL) [65, 107] has been demonstrated to be an efficient distributed machine learning framework to train a joint model from decentralized data. Recently, it has been paid more attention in the research field because of the highly developed IoT applications [34]. FL provides a privacy-preserved learning framework to address distributed optimization problems by allowing communications of learning information between remote users in the network, instead of sharing the private training datasets. Typically, an FL system consists of two parts: remote clients and an aggregator. Each remote client manages its private training data and performs a local learning process to obtain learning model updates, and the aggregator repeatedly updates a joint model from the received remote learning updates with an aggregation rule.

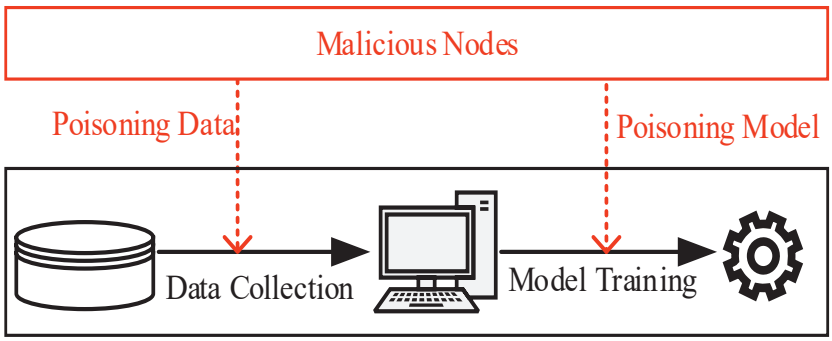


Figure 1.2: Data and model poisoning attacks on an FL system.

However, the distributed architecture of FL makes this learning system vulnerable to various attacks, as the remote clients could be easily compromised by attackers. Typically, the attacker can leverage the privacy property (i.e., the private remote training dataset) to intrude on several clients and manipulate their local training process, which leads to a decreased performance of the joint model. Figure 1.2 shows that the FL system is exposed to poisoning attacks at two stages: i) local data collection: data poisoning attacks can inject malicious data or modify existing data during the local data collection process; ii) remote model training: model poisoning attacks can directly inject poisoned parameters into the remotely trained model which is sent back to the aggregator. As such, poisoning attacks produce malicious updates to the aggregation process of FL. Note that, through careful manipulation, both data and model poisoning attacks can make the FL joint learning model converged so that the attackers are hard to be detected [11, 18, 108, 171, 44, 42, 17].

It is necessary to develop a defense method to protect the FL system against poisoning attacks. Typically, an FL defense method is considered to be successful if it can sanitize the poisoned remote updates to obtain a trusted joint model. One type of existing mechanisms [20, 185, 41, 12] is to detect malicious updates based on Euclidean distances or angle differences between each pair of remote updates [44, 13]. The other type of defense method is developed based on the Byzantine tolerance [20, 172]. However, recent studies have shown that poisoning attacks with constrained malicious data have the potential to bypass existing

defense methods [108, 151]. We notice that most of the existing defense approaches only regard the malicious updates as the global anomaly to the FL system, and they do not analyze local feature patterns of the malicious remote updates. In this dissertation, we introduce a new defense method based on a local feature analysis strategy: the maliciousness of poisoned remote updates is evaluated according to their model parameter features.

We propose our two-phase defense algorithm Local Malicious Factor (LoMar), which is able to detect the anomalies in FL from a local view, instead of the existing global view. The main idea of the proposed LoMar is to evaluate the remote update maliciousness based on the statistical characteristic analysis of the model parameters, which is intuitively motivated by the fact that each remote update in the FL system can be considered as being generated from a specific distribution of the parameters. Specifically, once the aggregator receives remote updates from a client, instead of using the whole remote updates set, LoMar performs the feature analysis of this update with its nearest neighbors. To measure the degree of maliciousness, a non-parametric local kernel density estimation method is applied to measure the relative distribution of the remote update in its neighborhood. We evaluate our proposed LoMar defense algorithm via both theoretical analysis and comprehensive experiments. In summary, we highlight the contribution of this dissertation as follows: 1) We propose a new two-phase defense algorithm, called LoMar, in order to address poisoning attacks against FL. 2) The proposed LoMar defense algorithm measures the malicious degree of remote updates based on its neighborhood by analyzing the statistical model parameter features via a non-parametric relative kernel density estimation method. 3) Besides the provided theoretical analysis of LoMar, we conduct an extensive performance evaluation of LoMar under two categories of poisoning attacks on FL. The results show that the proposed LoMar outperforms existing FL defense algorithms.

1.2.2 How to Test the Randomness for Security?

Leveraging the wireless channel randomness has become one of the fundamental approaches to building low-cost security designs for emerging wireless applications, such as radio frequency identification (RFID) [157] and Internet of Things (IoT) [198]. In particular, two communication parties, Alice and Bob, can use the random yet reciprocal wireless channel measurements, such as received signal strength information (RSSI) [59, 90, 122], channel state information (CSI) [169] and phase shifts [160, 161], to generate a common secret sequence to build a security design, such as secret key generation [103, 59, 156, 26, 90, 160, 161, 89, 154, 198, 1, 122, 169], secure communication [85, 94] and user authentication [4]. Then, Alice and Bob can enter the cryptographic domain [59, 90, 122] and use information reconciliation [21] and privacy amplification [16, 104] to compress their respective bit sequences. The framework is considered secure enough because a third-party eavesdropper Eve is expected to gain little information about the shared secret if she is more than half the wavelength away from them because of wireless fading [103].

To evaluate the security, existing studies [103, 59, 156, 26, 90, 160, 161, 89, 154, 198, 1, 122, 169] adopt the NIST randomness tests [14] to evaluate whether the underlying secret bit sequences generated from the wireless channel exhibit randomness properties, which has become the common practice. However, many of them [103, 59, 90, 26, 169] simply adopt the default NIST choices to set up a randomness test and consider successfully passing the test as a demonstration of security strength. Although passing a randomness test may hint that there is no major flaw in the design, it may not provide a guaranteed level of security strength. It becomes necessary to quantitatively understand the security impact of setting up randomness testing for the designs extracting random secrets from the wireless channel.

At the same time, efficiency is always another important design aspect. In secret key generation [59, 90, 26], efficiency refers to the key generation rate, which depends on the strictness of the randomness test and the sequence compression rate for privacy amplification. However, there is no theoretical analysis in the literature on how to guarantee efficient

secret key generation. Therefore, it is necessary to provide a design guideline for secret key generation to ensure both security and efficiency.

In this dissertation, we ask a fundamental question: How to properly set up statistical randomness tests for testing the wireless channel for both security and efficiency? In the current study, the mismatched assumption between practical channel coherence and theoretical memoryless assumption leads to a gray area in realistic wireless key establishment applications. Each test in NIST assumes that the bit sequence is IID, but in a practical wireless communication framework, the RSSI or CSI cannot be considered as IID [106, 143]. Despite the correlation, it can also pass the NIST tests by choosing $\alpha = 0.01$ [103, 59, 90] and 0.05 [26, 169].

Before we verify the security level of any secret key generation design, the first step is to clearly define an adversary model. A formal adversary model will enable thorough security analysis and evaluation, but it has not been systematically studied in the literature. To this end, we study how an adversary can defeat a security design by taking advantage of a potential defect of the wireless channel, which is unpredictable with unknown ground truth. In particular, it is never known whether a channel can indeed yield independently random sequences for secret key generation. Therefore, under the assumption that the generated secret sequence is indeed statistically correlated, an adversary can search for it from the most-likely sequence candidate to the least-likely one, as opposed to random guessing. This strategy, called Maximum Likelihood Tree Search (MLTS), is considered for the security evaluation of a wireless channel randomness-based design.

With the understanding of the adversary's capability, we propose a new design guideline for randomness testing, which involves solving an optimization problem that maximizes key generation efficiency under guaranteed security. In particular, we derive a mathematical formula for choosing the proper P-value threshold for nine different NIST tests. To our best knowledge, our design guideline is the first theoretical framework for wireless channel randomness-based secret key generation, where the randomness test parameters are not

empirically set. We note that rather than designing a new secret key generation method from the wireless channel, our focus is on how to properly set up the randomness tests. We conduct real-world experiments to validate the analysis, incorporate our design guideline into seven popular key generation methods and compare the difference. The results show that (i) using our design guideline, these methods achieve zero security loss in various experiment scenarios; (ii) the key generation efficiency can be significantly improved; (iii) Our design guideline is more adaptive for generating different bit lengths of key sequences. In summary, the main contributions of this paper are as follows: 1) We introduce the MLTS strategy to formalize the security analysis and evaluation of wireless channel randomness-based designs. 2) We propose a new design guideline on how to properly set up the randomness test parameter to eliminate security loss and achieve high efficiency. 3) We conduct the experiments in practical environments to show the improvement of our design guideline compared with existing secret key generation studies.

1.3 Dissertation Overview

In Chapter 2, we present a novel multi-server FL network, which can leverage the current edge computing system to sufficiently reduce the communication burden. In addition, we propose an FL algorithm called MS-FedAvg, which can bridge the information across all clients by the clients located in overlapping areas. In Chapter 3, we present a new online client-selection policy on HFL architecture, called COCS, which outperforms the existing studies and achieves sub-linear regret. In Chapter 4, we first analyze the targeted local poisoning attack on FL systems. Then, we propose the defense strategy, called LoMar, which leverages the kernel method and successfully protects the FL system even though there are large amounts of malicious clients. In Chapter 5, we analyze the insufficiently random channel response and point out the degradation of existing wireless key establishment strategies. Based on our analysis, we propose a guideline for how to generate a secret key on a wireless channel for security.

Chapter 2: On the Convergence of Multi-Server FL with Overlapping Area

2.1 Abstract

Multi-server Federated learning (FL) has been considered a promising solution to address the limited communication resource problem of single-server FL. We consider a typical multi-server FL architecture, where the coverage areas of regional servers may overlap. The key point of this architecture is that the clients located in the overlapping areas update their local models based on the average model of all accessible regional models, which enables indirect model sharing among different regional servers. Due to the complicated network topology, the convergence analysis is much more challenging than in single-server FL. In this chapter, we first propose a novel MS-FedAvg algorithm for this multi-server FL architecture and analyze its convergence on non-iid datasets for general non-convex settings. Since the number of clients located in each regional server is much less than in single-server FL, the bandwidth of each client should be large enough to successfully communicate training models with the server, which indicates that full client participation can work in multi-server FL. Also, we provide the convergence analysis of the partial client participation scheme and develop a new biased partial participation strategy to further accelerate convergence. Our results indicate that the convergence results highly depend on the ratio of the number of clients in each area type to the total number of clients in all three strategies. The extensive experiments show remarkable performance and support our theoretical results[1].

[1]This chapter was accepted in IEEE Transactions on Mobile Computing, August 2022. Permission is included in Appendix D.

2.2 Preliminary of FL

We consider a FL network including a number of N clients, indexed by $\mathcal{N} = \{1, ..., N\}$ and one central server/aggregator, where each client $i \in \mathcal{N}$ has its own local dataset with the data distribution D_i. FL aims to solve the following risk minimization problem:

$$\min_{\mathbf{w}} \left\{ f(\mathbf{w}) \triangleq \frac{1}{N} \sum_{i=1}^{N} F_i(\mathbf{w}) \right\}, \tag{2.1}$$

where $F_i(\mathbf{w}) \triangleq \mathbb{E}_{\xi \sim D_i}[F_i(\mathbf{w}, \xi)]$ is the local loss function. FedAvg [107], a seminal FL algorithm, works in an iterative manner as follows:

1) In each communication round t, each client i downloads the current global model $\mathbf{w}^t$ from the server and sets its initial local model as the current global model, i.e., $\mathbf{w}_i^{t,0} = \mathbf{w}^t$.

2) Each client runs E steps of Stochastic Gradient Descent (SGD) as follows:

$$\mathbf{w}_i^{t,e+1} = \mathbf{w}_i^{t,e} - \eta_l \nabla F_i(\mathbf{w}_i^{t,e}), \forall e = 0, ..., E-1, \tag{2.2}$$

where η is the learning rate of local training. Client i's updated model after these E steps can be written as $\mathbf{w}_i^{t+1} = \mathbf{w}_i^{t,E}$.

3) Each client uploads the updated model $\mathbf{w}_i^{t+1}$ to the server, which computes a simple aggregation $\mathbf{w}^{t+1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{w}_i^{t+1}$.

Due to the computation and bandwidth limitation, full client participation is often not practical. Hence, the more realistic FL strategy is that the server can select a subset of K clients, indexed by $\mathcal{K}^t \subseteq \mathcal{N}$, to participate in FL in communication round t, and the global model is computed according to $\mathbf{w}^{t+1} = \frac{1}{K} \sum_{i \in \mathcal{K}^t} \mathbf{w}_i^{t+1}$. This is known as partial client participation strategy [80, 181, 62].

2.3 Multi-Server FedAvg (MS-FedAvg)

A single-server FL system may incur a large delay if clients are distributed in the network, some of which may be far away from the server. Developing a multi-server FL network

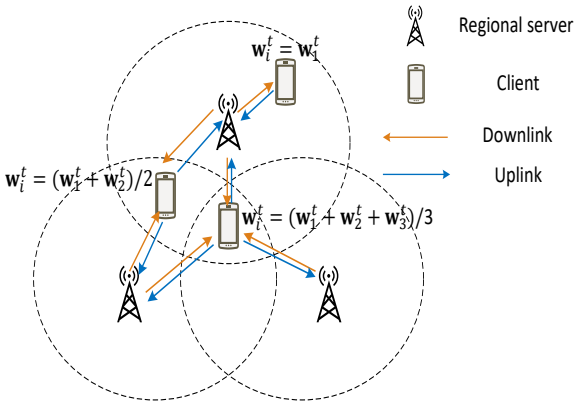


Figure 2.1: Description of MS-FedAvg.

architecture is a potential way to address this problem, e.g., Hierarchical FL (HFL) [86, 92, 159] and Clustered FL (CFL) [174, 74, 38]. While HFL requires the models on the edge server should be aggregated on the global server every several communication rounds, it also incurs extra transmission delay. CFL needs to re-cluster the clients every communication round based on a specific rule, e.g., model similarity or location, it is difficult to avoid the clients being far away from the edge server which should be clustered.

Since all the existing multi-server FL network architectures cannot directly leverage solve the large delay problem, we consider a multi-server FL architecture as in [50], where multiple regional servers are distributed close to the clients. Let $\mathcal{M} = \{1, ..., M\}$ be the set of regional servers and each regional server m covers a subset of client $\mathcal{N}_m \subseteq \mathcal{N}$ with $|\mathcal{N}_m| = N_m$. For convenience, we call $\mathcal{N}_m$ region m. It is worth noting that a client may locate in multiple regions, because the coverage areas of the servers may overlap.

In MS-FedAvg, each regional server trains a regional model using clients in its region, and a client updates its local model based on all regional models that it can access, where the architecture with three regional servers is shown in Figure 2.1. Different from HFL, the regional models are not aggregated until the final round to generate a global model. Let $\mathcal{M}_i \subseteq \mathcal{M}$ be the set of regional servers that client i can communicate where $M_i = |\mathcal{M}_i|$, and $\mathcal{M}_i^t \subseteq \mathcal{M}$ be the set of servers that client i is sampled in communication round t.

At the beginning of a communication round t, any client i downloads the current regional models $\mathbf{w}_m^t, \forall m \in \mathcal{M}_i$ from all the M_i servers, and averages the downloaded regional models to be the initial local model in the current round, i.e., $\mathbf{w}_i^{t,0} = \frac{1}{M_i} \sum_{m \in \mathcal{M}_i} \mathbf{w}_m^t$. Then, each client updates its local model using SGD for E local training epochs to obtain the local model $\mathbf{w}_i^{t+1}$ by (2.2), and uploads it to the servers in $\mathcal{M}_i^t$. Each server m then updates the regional model according to $\mathbf{w}_m^{t+1} = \frac{\eta_g}{N_m} \sum_{i \in \mathcal{N}_m^t} \mathbf{w}_i^{t+1}$, where η_g is the regional learning rate. After a sufficient number of T communication rounds, the global model is finally obtained by averaging over the converged regional models, i.e., $\mathbf{w} = \frac{1}{M} \sum_{m \in \mathcal{M}} \mathbf{w}_m^T$.

Compared to the single-server FL [66, 107, 80, 62], a unique feature of MS-FedAvg is that clients in overlapping areas receive average multiple regional models to be the initial model for local training in each communication round (Line 10). Together with the model averaging at the servers, this two-sided model averaging process allows the servers to indirectly access the local models of clients outside their regions instead of combining local model updates from the clients in overlapping areas (Line 13), which bridges the regional model sharing, thereby fully utilizing all clients' data in the network. Specifically, optimizing the placement of the limited number of regional servers in the current mobile computing system to maximize the total coverage is considered rather important. For example, some popular ES placement algorithms [72, 33] have shown that one overlapping area at most includes four regional servers. Therefore, we consider that the additional transmission latency of the overlapping areas clients mentioned in the chapter can be very small and negligible. Although the clients are located in overlapping areas, we can leverage the broadcast technique that cannot incur a high burden of communication.

Since we consider that the regional server is similar to the edge server, where the coverage is very restricted [29], the number of clients in the region of each regional server is much fewer than in single-server FL. Therefore, the communication burden is less than in single-server FL due to the shorter distance and more stable connection, and hence full client participation should work well in our proposed multi-server FL architecture. In this chapter, we only

consider the location of clients as fixed for proving the convergence results of MS-FedAvg. The static scenario can be considered as the hospital data [175] or environmental monitoring sensors [95], where the clients cannot move and only connect to the corresponding regional server(s). As such, our MS-FedAvg can improve this scenario efficiently. If the clients move randomly, each regional model can be assumed as an individual FL model approaching by FedAvg, which might degrade the training performance. In Section 2.7, we propose the experimental results of the movement scenario under our multi-server FL architecture. More specifically, [68, 114] developed the FL-based license plate recognition and human activity recognition algorithm. Firstly, the recognition results can be quickly obtained due to the less transmission latency. On the other hand, when clients come into overlapping areas, multiple monitors can acquire more information and recognize them more accurately.

Because the regional models are averaged only in the final communication round, a significant amount of communication and computation costs among the regional servers can be saved. However, the model averaging at the client side also introduces an obvious difference compared to single-server FedAvg: the initial models of the clients, even for those in the same region, for local training in each communication round can be different depending on their specific locations. Since we consider that the regional server is an edge server and the coverage area is very restricted, the number of clients in the region of each regional server is much fewer than in single-server FL. Therefore, the communication burden is less than in single-server FL, and hence full client participation should work well in our proposed multi-server FL. To clarify the reduction of transmission latency compared to other FL architectures, we will show the detailed quantification in Section 2.6.

In this dissertation, we also consider the partial client participation strategy, which is a more realistic strategy for single-server FL [107, 80, 62, 181]. More specifically, at the beginning of a communication round t, each server m randomly samples a subset of clients $\mathcal{K}_m^t \subseteq \mathcal{N}_m$ in its region to participate in the current round's training, with $K_m = |\mathcal{K}_m^t|$. Because a client may be in multiple regions, it may be sampled by multiple servers, which brings

more challenges for convergence analysis compared to single-server FL. We also provide the convergence analysis of the partial client participation strategy of MS-FedAvg.

2.4 Convergence Analysis of MS-FedAvg

In this section, we focus on a representative region $\mathcal{N}_m$ and study the convergence of its regional model. Let $f_m(\mathbf{w}) = \frac{1}{N_m}\sum_{i\in\mathcal{N}_m} F_i(\mathbf{w})$ be the objectives of region m. As discussed in the last section, the main difficulty of the convergence analysis lies in the heterogeneous initial models of clients in the region in each communication round. The convergence analysis encompasses non-iid datasets for general non-convex loss settings under both full and partial client participation strategies. Besides the existing random participation, i.e., unbiased client participation [79, 80, 78, 62, 146], we also propose a new biased client participation strategy for the MS-FedAvg algorithm. To propose convergence results of MS-FedAvg, we first state some useful assumptions in this chapter as follows:

Assumption 1. *(Lipschitz Gradient)* $\forall i \in \mathcal{N}_m$, F_i *is* L*-smooth, i.e., for all* $\mathbf{v}$ *and* $\mathbf{w}$, $F_i(\mathbf{v}) \leq F_i(\mathbf{w}) + (\mathbf{v}-\mathbf{w})^T\nabla F_i(\mathbf{w}) + \frac{L}{2}\|\mathbf{v}-\mathbf{w}\|_2^2$.

Assumption 2. *(Unbiased Local Gradient Estimator) Let* ξ_i *be a random local data sample on client* i. $\forall i \in \mathcal{N}_m$ *and* $\forall \mathbf{w}$, *the local gradient estimator is unbiased, i.e.,* $\mathbb{E}[\nabla F_i(\mathbf{w}, \xi_i)] = \nabla F_i(\mathbf{w})$, *where the expectation is taken over all the local datasets samples.*

Assumption 3. *(Bounded Local Variance)* $\forall i \in \mathcal{N}_m$ *and* $\forall \mathbf{w}$, *the variance of local gradient estimator of any regional server* m *can be upper-bounded by a constant* σ_m, *i.e.,* $\mathbb{E}\|\nabla F_i(\mathbf{w}, \xi_i) - \nabla F_i(\mathbf{w})\|^2 \leq \sigma_m^2$.

Assumptions 1-3 are fairly standard in existing FL works [80, 186, 125]. For the following assumption, we need to introduce the notion of the type of a client. Even for clients in the same region, they differ in terms of the subset of servers they may access since they may be in different overlapping coverage areas. Thus, we say that two clients have the same type if they can access the same set of servers. Formally, we define the client type $\theta \subseteq 2^{\mathcal{M}}$ to be the

subset of servers that it can access. Let $\mathcal{K}_{m,\theta}^t$ be the set of clients of type θ that is sampled in region m in round t, and let $K_{m,\theta}^t = |\mathcal{K}_{m,\theta}^t|$. Clearly, for all clients in region m, m must be an element of their types. Moreover, if two regions m and m' do not overlap, then there must be no client whose type contains both m and m'.

Assumption 4. *(Bounded Regional Variance) For any client i of type θ in region m and for any round t, the gradient difference of its local loss function at $\mathbf{w}_i^{t+1}$ and the regional loss function at $\mathbf{w}_m^{t+1}$ is upper-bounded, i.e.,* $\|\nabla F_i(\mathbf{w}_i^{t+1}) - \nabla f_m(\mathbf{w}_m^{t+1})\|^2 \leq \alpha_{m,\theta}^2$.

Assumption 4 states that clients of different types have different impacts on the gradient of the regional loss function at the end of each round. This impact is a joint result of the non-iid local datasets and different initial model at the beginning of the training round due to different coverage areas, i.e., types, which is different from the single-server FL [181, 132]. It is worth noting that in this chapter we do not assume to bound gradient descent [80, 50], i.e., $\|\nabla f(\mathbf{w})\|^2 \leq G^2$, where it is a loose assumption.

Before analyzing the convergence results of MS-FedAvg algorithm, we first propose the key lemma for both full and partial client participation strategies, which aims to propose the upper-bound of client drift for every regional model.

Lemma 1. *For any $\eta_l < \frac{1}{\sqrt{30}LE}$, we have the following results:*

$$\frac{1}{N_{m,\theta}} \sum\nolimits_{i \in \mathcal{N}_{m,\theta}} \mathbb{E}\|\mathbf{w}_i^{t,e} - \mathbf{w}_m^t\|^2 \leq 5E\eta_l^2 \left(\sigma_m^2 + \frac{6EN_{m,\theta}}{N} \alpha_{m,\theta}^2 \right) + 30E^2\eta_l^2 \|\nabla f(\mathbf{w}_m^t)\|^2.$$

Proof. The proof is shown in Appendix A.1. □

2.4.1 Analysis for Full Client Participation

For the full client participation of MS-FedAvg, we have the following convergence result:

Theorem 1. *Let assumptions 1-4 hold and $L, \sigma_m^2, \alpha_{m,\theta}^2$ be defined therein. With full client participation strategy, if we choose the learning rate $\eta_l \leq \min\{\frac{1}{\sqrt{30}LE}, \frac{1}{LE\eta_g}\}$. The convergence*

result is given as $\min_{t\in[T]} \mathbb{E}\|\nabla f(\mathbf{w}^t)\|^2 \leq \frac{f^0-f^*}{cMET\eta_g\eta_l} + \Psi$, *where* c *is a constant,* $f^0 \triangleq f_m(\mathbf{w}^0)$, $f^* \triangleq f(\mathbf{w}^*)$,

$$\Psi = \frac{1}{c}\sum\nolimits_{m\in\mathcal{M}}\left[\frac{L\eta_g\eta_l}{2MN_m}\sigma_m^2 + \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}}\frac{5N_{m,\theta}EL^2\eta_l^2}{2MN_m}\left(\sigma_m^2 + \frac{6EN_{m,\theta}}{MN_m}\alpha_{m,\theta}^2\right)\right].$$

Proof. The proof is shown in Appendix A.2. □

Remark 1. For the full client participation strategy of MS-FedAvg algorithm, the convergence rate has two parts: a vanishing term $\frac{f_0-f_*}{cMET\eta_g\eta_l}$ with increasing T and a constant term Ψ. The first part of Ψ, i.e., $\frac{L\eta_g\eta_l}{2MN_m}\sigma_m^2$, comes from the local stochastic gradient variance of each client, which shrinks when N_m increases. The cumulative variance of E local training contributes to the second term of Ψ, which has two variances and is largely affected by variance of different regional models $\alpha_{m,\theta}^2$.

Remark 2. The difference between MS-FedAvg and single server FedAvg [132, 181] comes from the term $\alpha_{m,\theta}^2$. Since the initial learning models $\mathbf{w}_i^t$ are the same for all clients, $\alpha_{m,\theta}^2$ is only related to the non-iid distribution of local datasets, i.e., $\alpha_{m,\theta}^2 = \alpha_m^2$, and the weight is all the same $\frac{1}{N_m}$. In MS-FedAvg, we observe that the contribution of $\alpha_{m,\theta}^2$ depends on the number of clients in each area type θ, i.e., $\frac{6EN_{m,\theta}^2}{N_m^2}\alpha_{m,\theta}^2$. Intuitively, local model from clients of the area type with the most clients can dominate the regional model $\mathbf{w}_m^t$. Inspired by [181], to make Ψ small, we can set the local learning rate η_l inversely proportional to the number of local training epochs E, i.e., $\eta_l = O(\frac{1}{E})$.

To make the Theorem 1 more readable, we will simplify the result to the following convergence rate by properly choosing the learning rates η_g and η_l:

Corollary 1. *Suppose* η_g *and* η_l *satisfy the condition in Theorem 1. Let* $\eta_g = \sqrt{EN_m}$ *and* $\eta_l = \frac{1}{\sqrt{TEL}}$. *For sufficiently large* T, *the convergence rate of MS-FedAvg under full client participation strategy satisfies:*

$$\min_{t\in[T]} \mathbb{E}||\nabla f(\mathbf{w}^t)||^2 = O\left(\frac{1}{M}\sum\nolimits_{m\in\mathcal{M}}\left(\frac{1}{\sqrt{N_mET}} + \frac{\sigma_m^2}{ET} + \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}}\frac{N_{m,\theta}^2\alpha_{m,\theta}^2}{N_m^2}\frac{1}{T}\right)\right).$$

2.4.2 Analysis for Unbiased Partial Client Participation

Due to the limited resource for current FL wireless networks, a partial participation strategy (only part of the clients join into the current communication round) has been considered more practical than full participation in existing FL studies [80, 62, 125]. Also, partial participation can accelerate the training by neglecting stragglers. We also consider the same two sampling schemes [80, 125, 181] for MS-FedAvg algorithm, i.e., with/without replacement clients sampling schemes, where $\mathcal{K}_m^t$ is randomly sampled. Due to the random property, we call it unbiased client participation of our proposed multi-server FL in this chapter. More specifically, it is worth noting that the unbiased client participation strategy for MS-FedAvg implies that $\frac{\mathbb{E}[K_{m,\theta}]}{K_m} = \frac{N_{m,\theta}}{N_m}$. Then, we have the following convergence results:

Theorem 2. *Let assumptions 1-4 hold and* $L, \sigma_m^2, \alpha_{m,\theta}^2$ *be defined therein. Let* $\beta_{m,\theta}^2 = \sigma_m^2 + \frac{6EN_{m,\theta}\alpha_{m,\theta}^2}{N_m}$. *With the scheme I for the unbiased client participation strategy, if the learning rate is chosen as* $\eta_l < \min\left\{\frac{1}{\sqrt{30}EL}, \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{N_{m,\theta}(K_{m,\theta}-1)}{EL\eta_g K_m N_m}\right\}$ *and the condition* $30E^2L^2\eta_l^2 + \frac{L\eta_g\eta_l \sum_{\theta\subseteq 2^{\mathcal{M}}} N_{m,\theta}}{K_m N_m}(90E^3L^2\eta_l^2 + 3E) < 1$ *holds, the global model* $\mathbf{w}^T$ *generated by MS-FedAvg satisfies* $\min_{t\in[T]} \mathbb{E}||\nabla f(\mathbf{w}^t)||^2 \leq \frac{f^0 - f^*}{cM\eta_g\eta_l ET} + \Psi_1 + \Psi_2 + \Psi_3$, *where* c *is a constant,* $f^0 \triangleq f(\mathbf{w}^0)$ *and* $f^* \triangleq f(\mathbf{w}^*)$,

$$\Psi_1 = \sum\nolimits_{m\in\mathcal{M}} \frac{EL\eta_g\eta_l}{2cMK_m}\sigma_m^2, \Psi_2 = \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{3EL\eta_g\eta_l N_{m,\theta}}{2cMK_m N_m}\alpha_{m,\theta}^2$$
$$\Psi_3 = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \left(\frac{5N_{m,\theta}EL^2\eta_l^2}{2cMN_m} + \frac{15N_{m,\theta}E^2L^3\eta_g\eta_l^3}{cMK_m N_m}\right)\beta_{m,\theta}^2.$$

For the Scheme II, if the learning rate is $\eta_l < \min\left\{\frac{1}{\sqrt{30}EL}, \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{K_m^2 N_{m,\theta}(N_{m,\theta}-1)}{EL\eta_g N_m^2 K_{m,\theta}(K_{m,\theta}-1)}\right\}$ *and the condition* $30E^2L^2\eta_l^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{L\eta_g\eta_l(K_{m,\theta}-1)}{2K_m(N_{m,\theta}-1)}(90E^3L^2\eta_l^2 + 3E) < 1$ *holds, we obtain that*

$$\Psi_1 = \sum\nolimits_{m\in\mathcal{M}} \frac{L\eta_g\eta_l}{2cMK_m}\sigma_m^2, \Psi_2 = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{2EL^2\eta_g\eta_l N_{m,\theta}(N_{m,\theta}-K_{m,\theta})}{cMK_m N_m(N_{m,\theta}-1)}\alpha_{m,\theta}^2$$
$$\Psi_3 = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \left(\frac{5EL^3\eta_l^2 N_{m,\theta}}{2cMN_m} + \frac{15E^2L^3\eta_g\eta_l^3 N_{m,\theta}(N_{m,\theta}-K_{m,\theta})}{2cMN_m K_m(N_{m,\theta}-1)}\right)\beta_{m,\theta}^2.$$

Proof. The proof is shown in Appendix A.3. □

Similar to full participation, we restate the above result by properly choosing η_g and η_l:

Table 2.1: Convergence rate of existing benchmarks.

Algorithm	Network architecture	Convexity[1]	Assumptions[2]	Partial client	Convergence rate
FedAvg [80]	Single server	SC	BGD	✓	$O(\frac{E}{T})$
FedAvg [181]	Single server	NC	BGV	✓	$O(\frac{1}{\sqrt{NET}} + \frac{1}{T})$
MC-PSGD [36]	Cluster	SC	BGD; BMP	×	$O(\frac{1}{\sqrt{N_m T}})$
IFCA [45]	Cluster	SC	BGD	✓	$O(\frac{1}{\sqrt{N_m T}} + \frac{E}{T})$
HFL [159]	Hierarchical	NC	BGV; BLV	×	$O(\frac{1}{\sqrt{N_m T}} + \frac{1}{T})$
FedMes [50]	Multi-server with overlapping areas	SC	BGD	×	$O(\frac{KE^2}{N})$
MS-FedAvg	Multi-server with overlapping areas	NC	BGV	✓	$O(\frac{1}{\sqrt{N_m ET}} + \frac{1}{T})$

[1] Shorthand notation for convexity: SC: Strongly Convex and NC: Non-Convex.
[2] Shorthand notation for assumptions in the chapter: BGD is bounded gradient descent $\|\nabla f(\mathbf{w})\|^2 \leq G^2$; BGV is bounded global variance $\|\nabla f_i(\mathbf{w}) - \nabla f(\mathbf{w})\| \leq \sigma^2$; BMP is bounded model parameter $\|\mathbf{w}\|^2 \leq B^2$; BLV is bounded local variance $\|\nabla f_i(\mathbf{w}) - \nabla f_j(\mathbf{w})\|^2 \leq \epsilon^2$.

Corollary 2. *Suppose η_g and η_l satisfy the condition in Theorem 2. Let $\eta_g = \sqrt{EK_m}$ and $\eta_l = \frac{1}{\sqrt{T}EL}$. Then, for sufficiently large T, the convergence rate of MS-FedAvg under unbiased partial client participation strategy satisfies:*

$$\min_{t\in[T]} \mathbb{E}||\nabla f_m(\mathbf{w}_t)||^2 = O\Bigg(\frac{1}{M}\sum\nolimits_{m\in\mathcal{M}}\Bigg(\frac{1}{\sqrt{K_m ET}} + \frac{\sigma_m^2}{ET} + \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{N_{m,\theta}^2 \alpha_{m,\theta}^2}{N_m^2}\frac{\sqrt{E}}{\sqrt{K_m T}} + \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{N_{m,\theta}^2 \alpha_{m,\theta}^2}{N_m^2}\frac{1}{T}\Bigg)\Bigg).$$

Remark 3. The structure of the convergence rate of the MS-FedAvg algorithm under unbiased partial client participation strategy is similar to full client participation, except for an additional variance term Ψ_2. This indicates that the unbiased partial client participation strategy does not have a significant change in convergence except for an amplified variance due to fewer clients being sampled. Intuitively, it yields a good approximation of all the clients' datasets distribution in expectation.

Remark 4. From Corollary 2, we can see that the convergence rate of the unbiased partial participation strategy is not related to the number of clients in each area type $K_{m,\theta}$, but it highly depends on the ratio $\frac{N_{m,\theta}^2}{N_m^2}$. However, due to the complicated network topology of multi-server FL, clients in some area types may present extreme performance, e.g., large $\alpha_{m,\theta}^2$ to incur large training degradation. Hence, to further accelerate the convergence rate of MS-FedAvg, we will develop a new sampling strategy that samples different numbers of clients in different area types.

2.4.3 Analysis for Biased Partial Client Participation

Here, we aim to develop a new biased partial client participation to achieve a speedup of the MS-FedAvg algorithm. Let $\mathcal{K}_{m,\text{biased}}^t \subseteq \mathcal{N}_m$ be the sampled clients set based on this strategy with $K_m = |\mathcal{K}_{m,\text{biased}}^t|$. The main idea of this strategy is that the number of sampled clients $\mathbb{E}[K_{m,\theta}]$ in different area type θ is fixed, where the ratio $\frac{\mathbb{E}[K_{m,\theta}]}{K_m}$ may not be equal to $\frac{N_{m,\theta}}{N_m}$. $\frac{\mathbb{E}[K_{m,\theta}]}{K_m}$ reflects the degree of bias. Intuitively, we can reduce the sampling number $\mathbb{E}[K_{m,\theta}]$ for some area types with large $\alpha_{m,\theta}^2$ in order to reduce their convergence contribution. Note that this strategy also includes the same two schemes with/without replacement as the unbiased participation strategy. The convergence results are shown as follows:

Theorem 3. *Let assumptions 1-4 hold and $L, \sigma_m^2, \alpha_{m,\theta}^2$ be defined therein, and c is a constant. Let $\beta_{m,\theta}^2 = \sigma_m^2 + \frac{6EK_{m,\theta}}{K_m}$. With scheme I for the biased client participation strategy, if the learning rate is chosen as $\eta < \min\left\{\frac{1}{\sqrt{30}ET}, \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{K_{m,\theta}(K_{m,\theta}-1)}{EL\eta_g K_m^2}\right\}$ and the condition $30E^2L^2\eta_l^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{L\eta_g\eta_l K_{m,\theta}^2}{K_m^2 N_{m,\theta}}(90E^3L^2\eta_l^2 + 3E) < 1$ holds, the global model $\mathbf{w}^T$ generated by MS-FedAvg satisfies $\min_{t\in[T]} \mathbb{E}||\nabla f(\mathbf{w}^t)||^2 \leq \frac{f^0 - f^*}{cM\eta_g\eta_l ET} + \Psi_1 + \Psi_2 + \Psi_3$, where c is a constant, $f^0 \triangleq f(\mathbf{w}^0)$ and $f^* \triangleq f(\mathbf{w}^*)$,*

$$\Psi_1 = \sum\nolimits_{m\in\mathcal{M}} \frac{L\eta_g\eta_l}{2cMK_m}\sigma_m^2, \Psi_2 = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{3EL\eta_g\eta_l K_{m,\theta}^3}{2cMK_m^3 N_{m,\theta}}\alpha_{m,\theta}^2$$
$$\Psi_3 = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \left(\frac{5EL^2\eta_l^2 K_{m,\theta}}{2cMK_m} + \frac{15E^2L^3\eta_g\eta_l^3 K_{m,\theta}^2}{2cMK_m^2 N_{m,\theta}}\right)\beta_{m,\theta}^2.$$

For Scheme II, if the learning rates is $\eta_l < \min\left\{\frac{1}{\sqrt{30}KL}, \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{K_{m,\theta}^2(N_{m,\theta}-1)}{EL\eta_g N_{m,\theta}K_m(K_{m,\theta}-1)}\right\}$ and the condition $30E^2L^2\eta_l^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{L\eta_g\eta_l K_{m,\theta}(N_{m,\theta}-K_{m,\theta})}{K_m N_{m,\theta}(N_{m,\theta}-1)}(90E^3L^2\eta_l^2 + 3E) < 1$ holds, and then we obtain that

$$\Psi_1 = \sum\nolimits_{m\in\mathcal{M}} \frac{L\eta_g\eta_l}{2cMK_m}\sigma_m^2, \Psi_2 = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{3L\eta_g\eta_l K_{m,\theta}(N_{m,\theta}-K_{m,\theta})}{2cK_m^2 N_{m,\theta}(N_{m,\theta}-1)}\alpha_{m,\theta}^2$$
$$\Psi_3 = EL^2\eta_l^2 \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \left(\frac{5K_{m,\theta}EL^2\eta_l^2}{2cMK_m} + \frac{15E^2L^3\eta_g\eta_l^3 K_{m,\theta}(N_{m,\theta}-K_{m,\theta})}{2cMK_m N_{m,\theta}(N_{m,\theta}-1)}\right)\beta_{m,\theta}^2.$$

Proof. The proof is shown in Appendix A.4. □

Corollary 3. *Suppose η_g and η_l satisfy the condition in Theorem 2. Let $\eta_g = \sqrt{EK_m}$ and $\eta_l = \frac{1}{\sqrt{TEL}}$. Then, for sufficiently large T, the convergence rate of MS-FedAvg under biased partial client participation strategy satisfies:*

$$\min_{t\in[T]} \mathbb{E}||\nabla f_m(\mathbf{w}_t)||^2 = O\Bigg(\frac{1}{M}\sum\nolimits_{m\in\mathcal{M}}\Bigg(\frac{1}{\sqrt{K_m ET}} + \frac{\sigma_m^2}{ET} + \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{K_{m,\theta}^2\alpha_{m,\theta}^2}{K_m^2}\frac{\sqrt{E}}{\sqrt{K_m T}} + \sum\nolimits_{\theta\subseteq 2^{\mathcal{M}}} \frac{K_{m,\theta}^2\alpha_{m,\theta}^2}{K_m^2}\frac{1}{T}\Bigg)\Bigg).$$

Remark 5. From Corollary 3, we can see that the biased client participation strategy has the same structure as the unbiased strategy. The difference is that variances of $\alpha_{m,\theta}^2$ include the term $\frac{K_{m,\theta}^2}{K_m^2}$ not $\frac{N_{m,\theta}^2}{N_m^2}$. Obviously, it is not difficult to design $K_{m,\theta}$ for each area type to accelerate convergence, for example, we can sample more clients in some areas with lower $\alpha_{m,\theta}^2$ value (suppose that $\alpha_{m,\theta}^2$ is constant) in order to decrease the variance terms Ψ_2 and Ψ_3. More specifically, since the variance $\alpha_{m,\theta}^2$ should be related to $K_{m,\theta}$, i.e., increasing $K_{m,\theta}$ should decrease $\alpha_{m,\theta}^2$, if the sampling strategy is such a way, it achieves a significant speedup for convergence in training.

2.5 Discussion of MS-FedAvg

Based on the above results, we briefly discuss the theoretical analysis of MS-FedAvg and its implications.

1) Convergence rate: When T is sufficiently large compared to E, we can simplify the convergence rates $O(\frac{1}{\sqrt{N_m ET}} + \frac{1}{T})$ in Corollary 1 and $O(\frac{\sqrt{E}}{\sqrt{K_m T}} + \frac{1}{T})$ in Corollaries 2-3, which matches the rate in the general non-convex setting of single-server FL algorithms [62, 132, 181] without the consideration of transmission difference. Although some works proposed new algorithms for multi-server FL architectures [38, 74, 45, 174], few of them presented the detailed convergence analysis. In Table 2.1, we summarize the convergence rate of some existing FL studies. Compared to the convergence rate, it is easy to see that our proposed MS-FedAvg algorithm achieves linear speedup for general non-convex settings. More

specifically, our assumption is the most strict among these studies, and BMP assumption should be unrealistic.

2) Accuracy: Though theorem 3 shows that sampling clients from fewer area types can improve the convergence performance, if we miss the clients in some area types, the accuracy performance may be degraded due to overfitting. Therefore, the design condition is that $\mathbb{E}[K_{m,\theta}] > 0$, $\forall\theta$. Due to the complicated network topology of multi-server FL architecture, it is difficult to obtain the theoretical result of $K_{m,\theta}$. We will present the empirical results to support the results of the biased partial client participation strategy in the Section 2.7.

3) The number of local epochs E and client K_m: Our results show that the number of local training epochs can be set as $E \leq \frac{T}{K_m}$ to accelerate convergence. In addition, the local training epochs help the convergence by properly setting hyper-parameters, which match the previous results [107, 146, 181]. The results in Theorems 1-3 imply that the convergence rate can be improved by increasing the number of clients in each communication round.

4) Comparisons to FedMes [50]: Although the training procedure of MS-FedAvg is similar to FedMes in [50], the unique difference between these two algorithms is that our proposed MS-FedAvg can leverage the value of η_g, which has been demonstrated that finding an optimal η_g can accelerate the training performance [132, 181]. In Table 2.1, we can see that [50] only proposes the convex loss function of FedMes (e.g., logistic regression [80]). Although [50] presented the experimental results based on the CNN model and achieve improvement, it does not propose the theoretical analysis to support the result. Since most of existing machine learning algorithms are non-convex (e.g., CNN and LSTM [132, 181, 62]), the theoretical results in [50] is much more restricted. In this chapter, the theoretical analysis and experiments are both in general non-convex settings. In addition, the convergence analysis in [50] leverages the BGD assumption, which has been considered a loose assumption in existing FL studies [181]. As such, our convergence analysis is tighter than FedMes. Lastly, we propose two kinds of partial client participation strategies (each strategy has two sampling

schemes) and analyze the training performance based on the ratio of the number of clients in different area types, which did not mention in [50].

5) Limitations: The regional models in MS-FedAvg are not aggregated before the Tth round. Hence, the final round aggregation does not have a significant impact on the convergence. The implicit aggregation is because the clients in overlapping areas share information across all regional models. Considering all factors in MS-FedAvg, including architecture, client distribution, and heterogeneous local dataset contributes to the implicit aggregation to be captured difficultly so that the full analysis is mathematically intractable. Thus, we bound the factors that depend on the convergence results between different servers via Assumption 4, and analyze the convergence in each region. As such, the problem becomes tractable and at the same time does not substantially impact the final results. In the future, we will set the multi-server FL as a bipartite graph, and propose the consensus analysis (i.e., the convergence gap between regional models and global models).

2.6 Transmission Latency Analysis

1) MS-FedAvg: In the multi-server FL network, to calculate the running time $\tau_{\text{Multi}}(t)$ in every communication round t, we will present the expressions to compute the three main components local computing time $\tau_i^C(t)$, uploading time $\tau_{i,m}^U(t)$ and downloading time $\tau_{i,m}^D(t)$. Note that because our proposed algorithms mainly focus on the efficiency of transmission, and the local computing time $\tau_i^C(t)$ is negligible compared to transmission latency [66, 61], we omit this part in our experiments. In summary, the transmission latency $\tau_{\text{Multi}}(t)$ in each round t is the sum of the largest uploading time and downloading time, i.e.,

$$\tau_{\text{Multi}}(t) = \max_{i,m} \tau_{i,m}^U(t) + \max_{i,m} \tau_{i,m}^D(t). \tag{2.3}$$

Uploading time of client i in communication round t is defined as follows:

$$\tau_{i,m}^U(t) = \frac{q_i}{br_{i,m}^U(t)}, \tag{2.4}$$

where q_i is the data size of client i for uploading and $r_{i,m}^U(t)$ in bits/s/Hz denotes the uploading rate of client i to the corresponding regional server m in communication round t, which is defined as follows:

$$r_{i,m}^U(t) = \log_2\left(1 + \frac{p_{i,m}^U |g_{i,m}^U(t)|^2}{\mu^2}\right), \tag{2.5}$$

where $p_{i,m}^U$ is the uplink transmit power of and $g_{i,m}^U(t)$ is the uplink channel gain of client i to the corresponding regional server m in communication round t, and μ^2 is the channel noise. Note that b in Hz is the bandwidth of one channel, i.e., $b = B/N$, where B is the total bandwidth budget and N is the number of clients. If we use partial participation strategy $b = B/N$. Since our compared benchmarks include multiple different FL network architectures, bandwidth b is divided into three categories: (1) $b_{cr} = B_{cr}/N$ is the client to regional server bandwidth; (2) $b_{rc} = B_{rc}/N$ is the regional server to cloud server bandwidth and (3) $b_{cc} = B_{cc}/N$ is the client to cloud server bandwidth. In the real world mobile network, $b_{cr} \leq b_{rc} = b_{cc}$ [101].

The definition of downloading time of client i to the corresponding regional server m is $\tau_{i,m}^D(t)$ is similar to the uploading time $\tau_{i,m}^U(t)$, which is defined as $\tau_{i,m}^D(t) = \frac{q_{i,m}}{br_{i,m}^D(t)}$, where $r_{i,m}^D(t) = \log_2(1 + \frac{p_{i,m}^D |g_{i,m}^D(t)|^2}{\mu^2})$, $p_{i,m}^D$ is the downlink transmit power, $g_{i,m}^D(t)$ is the downlink channel gain of client i to the corresponding regional server m in communication t. Suppose that the total communication round to achieve the targeted testing accuracy is T_{Multi}, the total transmission time is $\tau_{\text{Multi}}^{\text{Total}} = \sum_{t=1}^{T_{\text{Multi}}} \tau(t)$. Specifically, the transmission latency calculation of FedMes [50] is the same as MS-FedAvg.

2) Single-server FL: In the single-server FL network architecture, all clients communicate to the central server to download/upload the model updates uploading/downloading. The transmission latency $\tau_{\text{Single}}(t)$ of the single-server FL for one communication round depends on the slowest client i, which is calculated by

$$\tau_{\text{Single}}(t) = \max_i \tau_i^U + \max_i \tau_i^D. \tag{2.6}$$

Note that the transmit power p_i^D and p_i^U and the channel gain g_i^D and g_i^U should decay with increasing the distance [88, 153]. The distance between clients and regional server(s) should be much less than the distance between clients and the central server. Even though many existing single-server FL studies have proposed the developed algorithm to improve the convergence rate [80, 62, 129], single-server FL also requires much more total transmission time due to the large value of $\tau_{\text{Single}}(t)$. The total transmission time of single-server FL is $\tau_{\text{Single}}^{\text{Total}} = \sum_{t=1}^{T_{\text{Single}}} \tau(t)$.

3) HFL: The HFL architecture includes both the regional servers and the central server [86, 92, 159], which has two aggregation schemes (i.e., edge aggregation and global aggregation). In each region aggregation round, each regional server aggregates the local model updates uploaded from the clients in its service area, where the transmission latency of region aggregation is

$$\tau_{\text{Region}}(t) = \max_{i,m} \tau_{i,m}^U(t) + \max_{i,m} \tau_{i,m}^D(t). \tag{2.7}$$

In the global aggregation round, the central server aggregates the model updates on each regional server in which the transmission latency is $\tau_{\text{Global}}(t) = \max_m \tau_m^U(t) + \max_m \tau_{i,m}^D(t)$.

Note that the global aggregation round is performed periodically at every t_{Global} edge aggregation round (i.e., $t_{\text{Global}} \geq 1$). Suppose that if HFL requires T_{Region} and T_{Global} to achieve the targeted accuracy, the total transmission time of HFL is $\tau_{\text{HFL}}^{\text{Total}} = \sum_{t=1}^{T_{\text{Region}}} \tau_{\text{Region}}(t) + \sum_{t'=1}^{T_{\text{Global}}} \tau_{\text{Global}}(t')$. We can observe that HFL has extra aggregation rounds (i.e., global aggregation round) compared to single-server FL and multi-server FL architectures from which $\tau_{\text{Global}} > \tau_{\text{Region}}$ due to the large distance between regional servers and the central server. More specifically, in Table 2.1, the convergence rate of HFL (i.e., $\frac{1}{\sqrt{N_m T}} + \frac{1}{T}$), which implies that HFL requires more communication round to achieve targeted accuracy and incurs large total transmission time.

4) CFL: We assume that the CFL architecture includes M regional servers, where the number of M is equal to the number of clusters. Although the calculation of one communication round transmission latency of CFL τ_{CFL} is the same as the multi-server FL in (2.3),

Table 2.2: Datasets and models.

Dataset	Task	Clients	Total samples	Batch size	Model
EMNIST [31]	Handwritten character recognition	85	81,425	16	2-layer CNN+2-layer FFN
CIFAR-10 [69]	Image classification	85	60,000	32	MobileNet-v2 [135]
CIFAR-100 [69]	Image classification	85	60,000	32	MobileNet-v2 [135]

the distance of clients to the regional server is usually larger than multi-server FL since the clustering policy aims to cluster the clients that perform similar dataset distribution [45, 39]. In addition, the capability of the regional server is much lower than central server, and hence the slowest client will high impact on the transmission latency (i.e., $\tau_{\text{CFL}} \gg \tau_{\text{Multi}}$). Specifically, CFL should re-cluster the clients after several communication rounds, which incurs extra communication latency. The CFL may incur high divergence of each cluster, which may degrade the convergence performance, and it should use more communication rounds to achieve the targeted accuracy. In summary, if the number of total communication round is $\mathcal{T}_{\text{CFL}}$ and the number of re-clustering is $\mathcal{T}_{\text{Cluster}}$, the total transmission time of CFL is $\tau_{\text{Total}}^{\text{CFL}} = \sum_{t=1}^{\mathcal{T}_{\text{CFL}}} \tau_{\text{CFL}}(t) + \sum_{t'=1}^{\mathcal{T}_{\text{Cluster}}} \tau_{\text{Cluster}}(t')$.

2.7 Experiments

In this section, we conduct multiple experiments to evaluate the performance of the proposed MS-FedAvg algorithm.

2.7.1 Experimental Setup

1) Datasets and models: We evaluate our proposed algorithms on three datasets: EMNIST [31], CIFAR-10 and CIFAR-100 [69]. In each dataset, we simulate the data heterogeneity by sampling the label ratios from a Dirichlet distribution with parameter 0.4 [53], and keep the training data on each client balanced. For the EMNIST dataset, we use the CNN model with two hidden layers and two FeedForward Network (FFN) layers, and the two learning rates are set as η_g =1.1 and η_l = 0.05 by grid search. For CIFAR-10 and CIFAR-100, we use MobileNet-v2 [135] to be the learning model, and the learning rates are

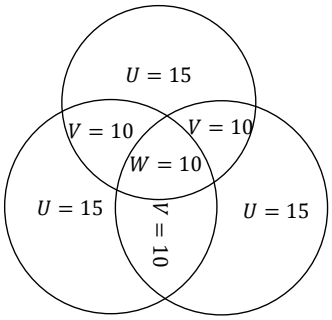


Figure 2.2: Symmetric multi-server FL architecture.

set as $\eta_g = 1.5$ and $\eta_l = 0.1$. Table 2.2 summarizes datasets, models, batch sizes, and the number of clients. All the hyper-parameters are set based on a grid search on each dataset. Note that all the algorithms are set $E = 5$ and K_m =10 by default.

2) Compared benchmarks: In this chapter, we compare our proposed algorithms to 5 existing FL benchmarks and can be concluded into 3 categories, i.e., single-server FL, HFL, and clustered FL. 1) FedAvg: FedAvg algorithm [107] is the most important baseline in FL research field. Note that the setting of FedAvg is the same as [181]. 2) Fedprox: Fedprox [79] develops a l_2-norm regularized algorithm to address the local model updates in the heterogeneous FL. In our experiment, we follow the settings in [79] with $\lambda = 0.01$, which controls the dissimilarity of local objectives. 3) HFL: HFL [93, 159] is a edge-cloud based FL architecture. In our experiment, we use one layer of edge servers, and after 5 times client to edge server communication rounds, edge servers upload the model to the cloud server to compute aggregation. 4) MC-PSGD: MC-PSGD [36] is a CFL architecture, which processes the local training by clustering the clients into several blocks to reduce the client drift. We assume that re-clustering the blocks in each communication round, and the re-clustering time τ_{Cluster} is 1/20 of one round. 5) IFCA: IFCA [45] is a clustered FL, which is clustered every 5 times communication round and based on calculating the cosine similarity, where τ_{Cluster} is 1/20 of one communication round. 6) FedMes: FedMes [50] is a multi-server FL, which sets the $\eta_g = 1$.

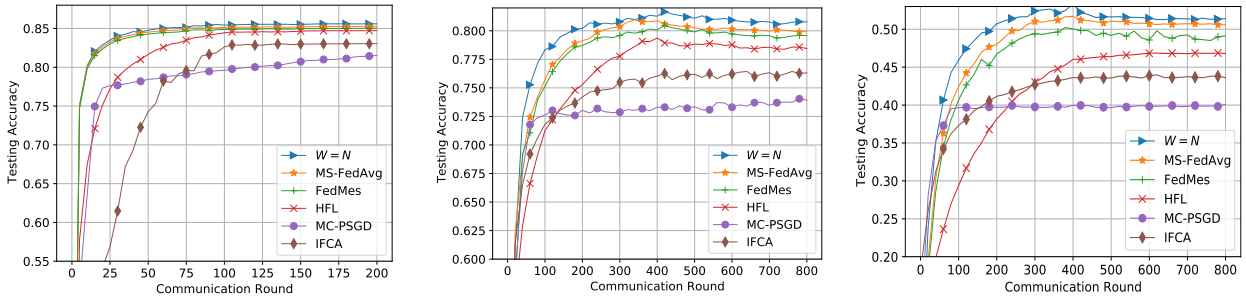


Figure 2.3: Full participation performance.

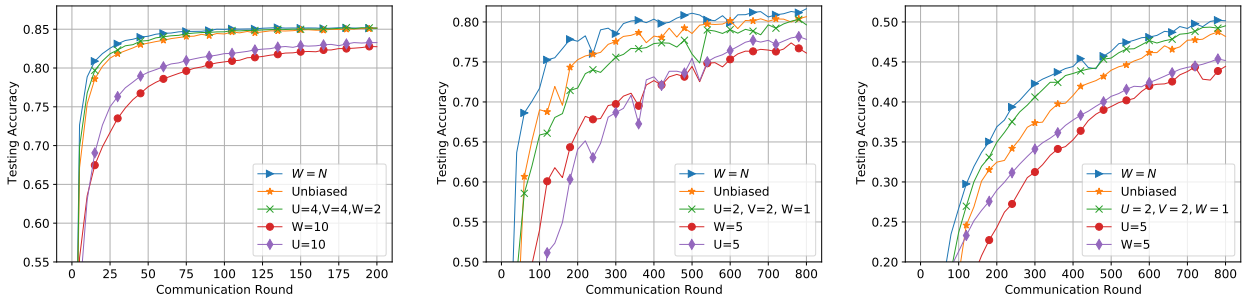


Figure 2.4: Partial participation performance.

Table 2.3: Final testing accuracy, round and wall-clock(sec).

Dataset	EMNIST			CIFAR-10			CIFAR-100		
Algorithm	Accuracy	Round	Wall-clock	Accuracy	Round	Wall-clock	Accuracy	Round	Wall-clock
FedAvg	85.06%	8	5.37	80.75%	88	1495.21	50.25%	95	2119.45
FedProx	84.97%	10	6.72	80.06%	106	1803.06	49.61%	110	2643.30
HFL	84.87%	31	16.12	77.52%	191	2555.58	46.19%	351	7563.98
MC-PSGD	83.03%	68	36.04	73.86%	NA	NA	NA	NA	NA
IFCA	83.85%	65	34.13	76.61%	291	4367.93	NA	NA	NA
FedMes	84.91%	16	7.64	79.08%	100	962.17	49.82%	130	2056.71
$W = N$	85.04%	11	5.97	80.98%	85	785.90	50.62%	90	1632.62
MS-FedAvg	85.02%	13	7.15	79.84%	91	903.17	50.14%	119	1959.72

3) Multi-server network architecture: We set our multi-server FL network architecture with $M = 3$ regional servers and 85 clients. Here, we consider a symmetric geometry multiple servers network with $U = 15$, $V = 10$ and $W = 10$ (U is the number of clients in the non-overlapping area for every server, V is the number of clients in the overlapping area between any two servers, W is the number of clients in the overlapping area among all three servers) such that each regional server covers 45 clients, which is shown in Figure 2.2. Another multi-server FL network is that all 85 clients are within in the overlapping area among all three servers and hence $W = 85$, $U = 0$ and $V = 0$. For the partial participation strategy, each

regional server randomly samples 10 clients in each communication round. The asymmetric network architecture will be presented later.

4) Network parameters setup: The network setting is summarized as follows unless otherwise specified. We consider the regional server with a disc of 2km and the cloud server with 5km. The channel gain of both the uplink and downlink are composed of both small-scale fading and large-scale fading. The small-scale fading is set as Rayleigh distribution with uniform variance and the large scale fading from client to regional server, client to cloud server and regional server to cloud server are all generated using the path-loss model $P_L = 128.1 + 37.6 \log_{10}(d(\text{km}))$, where d is the distance in km. The noise power μ^2 is -107dBm. Total bandwidth budget $B_{rc} = 850$MHz, $B_{cr} = 475$MHz and $B_{cc} = 150$MHz. Both the uplink and downlink transmit power is 23dBm, i.e., $p_i^U = p_i^D = 23$dBm, $\forall i \in \mathcal{N}$. These parameters are followed by the existing edge computing studies [153, 29, 40].

2.7.2 Performance Evaluations

Here, we mainly focus on comparing the performance with the multi-server FL benchmarks, including three settings: full client participation, partial client participation, and moving clients scenarios. Note that the setting $W = 85$ means that all 85 clients are located in the overlapping area with 3 regional servers, which is considered as the upper bound of MS-FedAvg since all three regional models reduce the divergence of the initial model.

1) Performance of full clients participation strategies: In Figure 2.3, we aim to show the performance of full clients participation strategy compared to the three different multi-server FL benchmarks. It is easy to see that our proposed MS-FedAvg algorithm converges faster and achieves the best accuracy performance than other benchmarks in all three datasets except for the setting with $W = 85$, which supports our theoretical results in Theorem 1. For example, in the CIFAR-10 dataset, MS-FedAvg can achieve 79.69% testing accuracy, which is 1.51%, 3.78%, and 5.59% higher than HFL, IFCA, and MC-PSGD. In particular, MC-PSGD converges fast but it achieves the lowest accuracy since the clustered model is

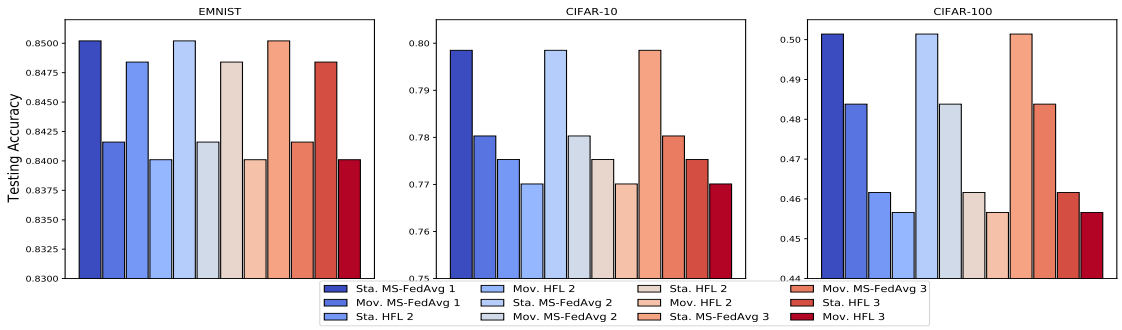


Figure 2.5: Testing accuracy under static and moving scenarios.

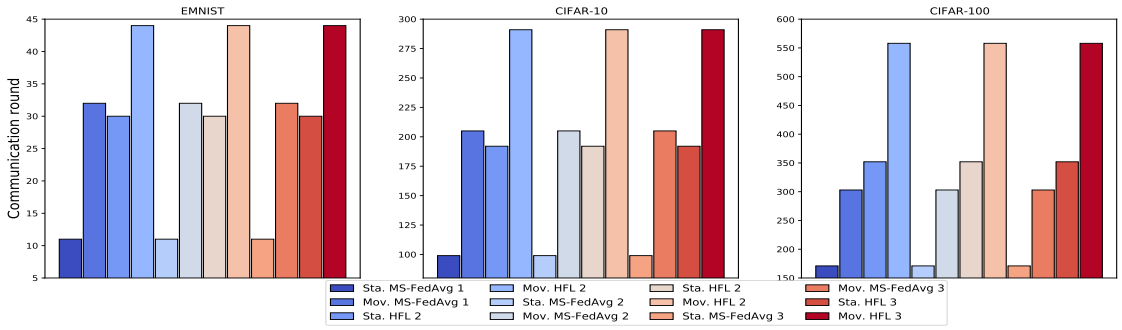


Figure 2.6: Communication rounds under static and moving scenarios.

easy to overfit to its cluster. However, the global model performance is the worst among all benchmarks, i.e., 74.10%. More specially, the disadvantage of HFL is that for every 5 regional aggregation steps, it is required a global aggregation, which may degrade all the regional learning performance. Although FedMes outperforms the other three benchmarks, it does not achieve a better convergence rate and accuracy than our proposed MS-FedAvg. The results may be because the value of η_g is not optimal.

2) Performance of partial client participation strategies: For the partial client participation strategy, we uniformly sample $K = 10$ clients in each region and communication round, and the performance of different values of K will be shown later. For convenience, the meaning of the legend is the number of sampled clients in each area type. U is the isolated area type, V is the clients located in the overlapping area with two regional servers and W is the three regional servers' overlapping area. For example, $U = 10$ is sampling 10 clients in area type U. Based on this network architecture, we have the following interesting observations.

Firstly, we can see that the unbiased partial participation of all three datasets in Figure 2.4 is similar to the performance of full client participation in Figure 2.3 but with higher variance, which is due to the uniformly sampling, and successfully matches our analysis in Section 2.5. If we select these 10 clients with the number of $U = 4$, $V = 4$, and $W = 2$, it performs better than the unbiased participation strategy, e.g., 1.43% higher testing accuracy than the unbiased MS-FedAvg in CIFAR-100 dataset. More specifically, if we only sample clients in one specific area, the performance has much degradation, e.g., 45.01% with $W = 10$, and 43.92% with $U = 10$. The reason may be that the regional model overfits these local clients and cannot be generalized to all clients in the entire FL network. Therefore, if we use a biased participation strategy, it is necessary to sample clients among all area types. The learning performance of Biased MS-FedAvg strongly depends on the network topology, and hence it is not easy to provide an optimized sampling strategy. However, it is feasible to find a sampling strategy that performs better than an unbiased strategy.

3) Performance of the client movement scenarios: Here, we aim to show the comparison of static and movement scenarios of multi-server FL settings. Because clustered FL needs to re-cluster every several communication rounds, the movement scenario can be ignored in this setting. Therefore, we only compare our proposed MS-FedAvg algorithm to HFL. Since we cannot justify the deterministic moving direction of each client, we assume that it randomly moves in each communication round. Because of the restricted service area of the regional server, we consider the movement scenario such as the moving sensors or IoT devices [163, 70], which performs low movement speed (e.g., 3 miles per hour [70]). Compared to the transmission latency, the moving distance is very short and we can assume that each client connects the same corresponding regional server(s) within one communication round. In order to evaluate the training performance between static and movement scenarios, we set three network settings: (1) the probabilities of the client locating in each area are $\mathbb{P}(\text{locate in } U) = 52.94\%$, $\mathbb{P}(\text{locate in } V) = 35.30\%$ and $\mathbb{P}(\text{locate in } W) = 11.76\%$; (2) $\mathbb{P}(\text{locate in } U) = 35.30\%$, $\mathbb{P}(\text{locate in } V) = 52.94\%$ and $\mathbb{P}(\text{locate in } W) = 11.76\%$; and (3)

$\mathbb{P}(\text{locate in } U) = 52.94\%$, $\mathbb{P}(\text{locate in } V) = 11.76\%$ and $\mathbb{P}(\text{locate in } W) = 35.30\%$. The communication round is to achieve 80% on the EMNIST dataset, 75% on the CIFAR-10 dataset, and 45% on the CIFAR-100 dataset.

We can see that the convergence rate of movement scenarios is much slower than static scenarios among all the multi-server FL settings, e.g., in the CIFAR-10 dataset, movement is 77.08% and static is 79.84% of MS-FedAvg. And it only uses 91 communication rounds to achieve 75% testing accuracy, which is much better than the movement scenario with 207 rounds. The reason is that since the clients may participate in different regional model training, it incurs higher model variance between each communication round. As a result, it makes the global model to be converged difficultly. It is similar to training the same several regional models on each regional server. Therefore, it is not necessary to consider the movement scenario in this chapter. In addition, it is clear to see that our proposed MS-FedAvg also outperforms other benchmarks in the movement scenarios. If we assume more clients locate in the overlapping areas (e.g., setting 3), the training performance of both MS-FedAvg and HFL improves. For example, in setting 3, the movement scenario of MS-FedAvg has 48.75% testing accuracy and uses 279 to achieve 45% accuracy on the CIFAR-100 dataset. This may be because the data distribution clients perform less diversity and are near the $W = N$ scenario. More specifically, MS-FedAvg outperforms HFL in all settings (e.g., on the CIFAR-10 dataset of setting 2, the movement scenario of MS-FedAvg achieves 78.01% and HFL is 76.18%).

2.7.3 Additional Performance

In this subsection, since it is not easy to verify the local computing time of each client, and the existing chapters have shown that [61, 107] the transmission latency dominates the running time FL, and hence we only compare the transmission latency to training performance of all FL benchmarks and simply ignore the local computing time of every client. The wall clock means that the total transmission time to achieve the targeted testing accuracy.

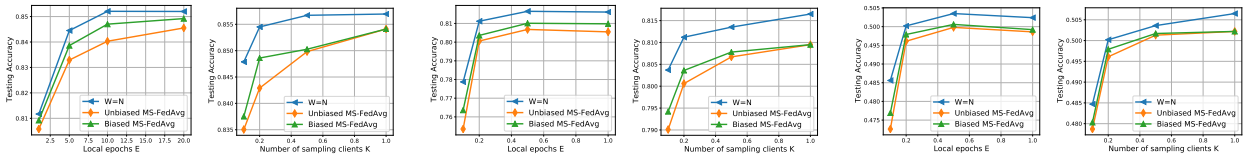


Figure 2.7: Impact on K_m and E.

Table 2.4: Impact on different bandwidth settings.

Dataset	EMNIST		CIFAR-10		CIFAR-100	
Bandwidth	Accuracy	Wall-clock	Accuracy	Wall-clock	Accuracy	Wall-clock
$b_{rc} = 10$MHz, $b_{cr} = 5$MHz	85.02%	7.15	79.84%	903.17	50.14%	1959.72
$b_{rc} \sim \mathcal{U}[8, 12]$MHz, $b_{cr} \sim \mathcal{U}[4, 6]$MHz	85.05%	10.49	79.59%	1003.26	49.98%	2227.01
$b_{rc} \sim \mathcal{U}[5, 15]$MHz, $b_{cr} \sim \mathcal{U}[2, 8]$MHz	84.97%	15.01	79.86%	1420.39	50.05%	2936.91

Table 2.3 shows the final testing accuracy, communication round, and wall-clock to achieve the targeted testing accuracy. We compare our MS-FedAvg algorithm to single-server FL, HFL, and CFL. It is easy to observe that the final testing accuracy of $W = N$ and FedAvg perform similarly among all three datasets, e.g., 78.98% and 78.96% in the CIFAR-10 dataset. This is because they do not have the model divergence to degrade the learning performance. The $W = N$ setting can be considered as the FedAvg on multi-server FL network architecture, while $W = N$ is much more efficient from the transmission latency perspective. For the EMNIST dataset, the reason that the FedAvg algorithm has the best performance is that the EMNIST dataset is simple and easy to achieve targeted testing accuracy. For the more complicated datasets, it is clear to see that MS-FedAvg outperforms single-server FL. FedMes outperforms the other three benchmarks, but it is worse than MS-FedAvg.

Although both the two single-server FL benchmarks perform with good accuracy performance, they will waste much more training time, due to the large average distance between clients and servers. Although the HFL and CFL algorithms spend less transmission latency for one communication round, they waste much more wall-clock time to achieve the targeted accuracy due to the low convergence rate, e.g., for the CIFAR-10 dataset, HFL uses 191 rounds and 2555.58 sec, and IFCA uses 291 rounds and 4367.93 sec. Therefore, the existing multi-server FL benchmarks cannot guarantee to be efficient enough from the transmission perspective.

Our proposed MS-FedAvg outperforms other multi-server FL benchmarks on testing accuracy perspectives. More specifically, MS-FedAvg has the best wall clock time among all the benchmarks except the $W = N$ setting, since it does not need to download and upload models to the central servers, it significantly reduces the distance and saves much more transmission latency. For example, in CIFAR-10 dataset, it saves 1.65×, 1.99×, 2.83× and 4.84× time than FedAvg, FedProx, HFL and IFCA. More specifically, due to the limited generalization of MC-PSGD, it cannot achieve the targeted accuracy. Therefore, our proposed MS-FedAvg algorithm can be considered an efficient solution to address the bottleneck problem of FL settings.

2.7.4 Impact on Different Parameters

1) Impact on K_m and E: Based on our analysis in Sections 2.4 and 2.5, the learning performance of MS-FedAvg algorithm depends on several hyper-parameters, e.g., the number of sampling clients under each regional servers K_m and the setting of the number of local epochs E. Figure 2.7 present the final testing accuracy of EMNIST, CIFAR-10 and CIFAR-100 datasets under different values of K_m and E. Especially, we set $2U = 2V = W$ as "Biased", which means the fraction of the number of sampling clients in different area types.

The results in Figure 2.7 indicate that the performance substantially improves when we increase the number of sampled clients number K_m, and the biased participation strategy consistently outperforms unbiased participation, e.g., in CIFAR-10 dataset, biased client participation strategy increases from 76.20% to 82.93%, when $K_m = 10$ and 35, and unbiased increases from 75.56% to 82.93%. In addition, the degree of improvement of K_m increases lower. This empirical result matches our analysis in Section 2.4, and performs similarly to single-server FL settings [80, 181, 62].

Next, we aim to show the learning performance under different values of E. Until now, it is difficult to explicitly show the relationship between E and learning performance. In [80], they presented that increasing E can improve performance. However, other studies [79, 181]

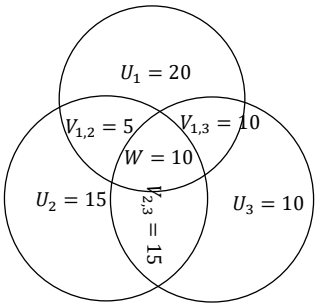


Figure 2.8: Asymmetric multi-server FL architecture.

showed that when E is set as too large, it will degrade the performance. Our experimental results in Figure 2.7 implies that if $E = 1$, it performs the worst. If we increase the value, the accuracy firstly increases but then decreases, e.g., in CIFAR-100, when $E = 5$, the accuracy is 49.65%, but 44.65% of $E = 20$. Thus, it is necessary to find a suitable value E to achieve better performance on different datasets.

2) Impact on bandwidth: Here, we present the impact on different bandwidth settings between clients and regional server(s), which includes three settings: (1) b_{rc} =10MHz, b_{cr} = 5MHz; (2) $b_{rc} \sim \mathcal{U}$[8, 12]MHz, $b_{cr} \sim \mathcal{U}$[4, 6]MHz; and (3) $b_{rc} \sim \mathcal{U}$[5, 15]MHz, $b_{cr} \sim \mathcal{U}$[2, 8]MHz, where $\mathcal{U}$ is uniform distribution.

In Table 2.4, we can see that different bandwidth does not have a significant impact on the testing accuracy. For example, on the EMNIST dataset, the testing accuracy of these three settings is 85.02%, 85.05%, and 84.97%. This is because the learning performance is independent of the network parameter settings, and only depends on the setting of learning models (e.g., data distribution and hyper-parameters). However, the bandwidth has a large influence on the transmission latency, because each regional server should wait for the slowest client that performs small bandwidth and then process the aggregation. On CIFAR-100 dataset, if the bandwidth follows $b_{rc} \sim \mathcal{U}$[5, 15]MHz, $b_{cr} \sim \mathcal{U}$[2, 8]MHz, the total communication time is 2936.91sec, which is 49.86% higher than equal bandwidth setting. Therefore, if each regional server has a limited bandwidth budget, the best way is to equally divide to each client, which can achieve the best performance on communication.

2.8 Related Work

Current research on FL wireless networks improved communication efficiency in the following aspects. (1) how to properly allocate resources to clients. [176] designs how to properly select clients and how bandwidth is allocated among the selected clients in each communication round. [140] jointly considers bandwidth allocation and client scheduling problems. For the bandwidth allocation sub-problem, they aim to allocate more bandwidth to the clients with worse channel conditions and develop a greedy policy to solve the clients scheduling sub-problem. (2) deadline-based FL architecture. [126] develops a client selection algorithm for deadline-based HFL via contextual combinatorial multi-armed bandits to improve the training performance. (3) physical layer quantization. For bandwidth reduction, [2] sparsifies the gradient estimates of clients to accumulate errors from previous communication rounds and project the resultant sparse into a low-dimensional vector. In [197], they clarify how to communicate between clients and the central server and evaluate the impact on the various quantization. In addition, they design the physical layer quantization both on uplink and downlink. They mainly minimize the communication latency by solving an optimization problem subject to the constraint of obtaining a good model. However, few of them propose the details of the convergence guarantee in their chapters.

FL was first proposed in [107], where they proposed the FedAvg algorithm and showed the advantages empirically on different datasets and local dataset distribution settings. Followed by [107], the authors propose a strategy to address the communication bottleneck problem by increasing the local training epochs [80, 181]. Specifically, this method is also a feasible solution to improve the convergence rate. Based on this method, some new algorithms are developed from different perspectives. [62] adds a variant control variable to reduce the local model updates and global model due to the non-iid distribution of local datasets, and [128] and [82] designs FL algorithm for asynchronuous FL via Hessian approximation. [132] designs server level momentum and extends the local SGD optimizer to AdaGrad, YOGI

and ADAM, [129, 177, 81] proposes client level momentum FL algorithm. However, these algorithms are mainly developed on single-server FL.

Based on the highly efficient edge computing architecture, some studies focus on edge facilitated FL: HFL [86, 92, 159] and clustered FL [174, 74, 38]. However, they also rely on communicating to the central server, large communication delay is difficult to be avoidable compared to our proposed multi-server FL. Another direction of distributed learning is fully decentralized/serverless [83, 134, 67]. In decentralized FL, clients need to exchange the model updates with their neighbors, not with the servers. This is different from our proposed multi-server FL architecture, where the local model updates are required to be aggregated on regional servers. However, even though the network of decentralized FL is well-connected, it is not avoidable to reduce the degradation due to large communication delays, since the bandwidth of each client should be much less than edge computing.

2.9 Summary

In this chapter, we proposed the MS-FedAvg algorithm and presented theoretical analysis on non-iid datasets in general non-convex settings on a multi-server FL architecture with overlapping areas, which can reduce transmission latency compared to traditional single-server FL. Our theoretical results reveal how the overlapping areas accelerate the convergence of the final global model. In addition, the MS-FedAvg algorithm achieves a linear speedup under full/unbiased partial client participation strategies compared to the existing multi-server FL algorithms. To further improve the convergence rate, we develop a biased partial client participation strategy. Both theoretical and empirical results show the degree of bias results in a trade-off between convergence rate and accuracy and outperforms other existing multi-server FL architectures. Although our work is based on the fundamental theory of traditional FL, it also opens doors to many new interesting questions in FL studies. For future work, we plan to investigate how to design the algorithms based on the topology of the multi-server FL architecture, and the consensus control.

Chapter 3: Context-Aware Online Client Selection for HFL

3.1 Abstract

Federated Learning (FL) has been considered an appealing framework to tackle data privacy issues of mobile devices compared to conventional Machine Learning (ML). Using Edge Servers (ESs) as intermediaries to perform model aggregation in proximity can reduce the transmission overhead, and it enables great potential in low-latency FL, where the hierarchical architecture of FL (HFL) has attracted more attention. Designing a proper client selection policy can significantly improve training performance, and it has been extensively used in conventional FL studies. However, to the best of our knowledge, no studies are focusing on HFL. In addition, client selection for HFL faces more challenges than conventional FL (e.g., the time-varying connection of client-ES pairs and the limited budget of the Network Operator (NO)). In this chapter, we investigate a client selection problem for HFL, where the NO learns the number of successful participating clients to improve training performance (i.e., select as many clients in each round) as well as under the limited budget on each ES. An online policy, called Context-aware Online Client Selection (COCS), is developed based on Contextual Combinatorial Multi-Armed Bandit (CC-MAB). COCS observes the side-information (context) of local computing and transmission of client-ES pairs and makes client selection decisions to maximize NO's utility given a limited budget. Theoretically, COCS achieves a sublinear regret compared to an Oracle policy on both strongly convex and non-convex HFL. Simulation results also support the efficiency of the proposed COCS policy on real-world datasets[2].

[2]This paper was published in IEEE Transactions on Parallel and Distributed Systems, vol. 22, pp. 4353-4367, December 2022. Permission is included in Appendix D.

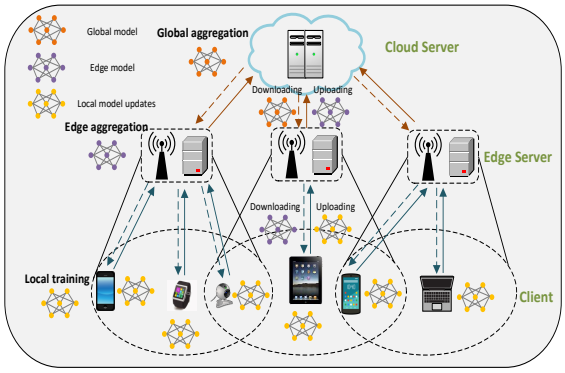


Figure 3.1: The architecture of HFL.

3.2 System Model and Problem Formulation

3.2.1 Preliminary of HFL

The Network Operator (NO) leverages a typical edge-cloud architecture to set a Federated Learning (FL) service, where it is named Hierarchical FL (HFL) [92, 93, 159] in Fig. 3.1. Unlike the conventional FL [107, 80, 62] only including clients and a Cloud Server (CS), HFL consists of a set of mobile devices/clients, indexed by $\mathcal{N} = \{1, 2, \dots, N\}$, a set of Edge Servers (ES), indexed by $\mathcal{M} = \{1, 2, \dots, M\}$ and a Cloud Server (CS). Let $\mathcal{N}_m^t = \{1, 2, \dots, N_m^t\}$ denote the set of clients, which can communicate with the ES m in edge aggregation round t. Note that the communication area of different edge servers may be overlapped (i.e., $\sum_{m=1}^{M} N_m^t \geq N$). The client $n \in \mathcal{N}$ is able to communicate to a subset of ESs $\mathcal{C}_n^t \subseteq \mathcal{M}$ in round t. In particular, we assume that each client is equipped with a single antenna such that it only communicates with one ES $m \in \mathcal{C}_n^t$ even if it is located in the overlapped area in each round. Let $\mathbf{w}$ denote the parameters of the global model. The goal of the FL service is to find the optimal parameters of global model $\mathbf{w}$, which minimizes the average loss function $f(\mathbf{w})$ under the HFL network as follows:

$$\min_{\mathbf{w}} f(\mathbf{w}) := \frac{1}{M} \sum\nolimits_{m \in \mathcal{M}} \frac{1}{S_m} \sum\nolimits_{n \in \mathbf{s}_m} F_n(\mathbf{w}), \tag{3.1}$$

where $\mathbf{s}_m$ is the selected client set by the ES m with the number S_m in each edge aggregation round, $F_n(\mathbf{w}) \triangleq \sum_{\xi_n \sim \mathcal{D}_n} \ell(\mathbf{w}; \xi_n)$ is the loss function associated with the local dataset $\mathcal{D}_n$ on

client n, and $\ell(\mathbf{w}; \xi_n)$ is the loss of data sample ξ_n. The objective of the loss function $F_n(\cdot)$ can be convex (e.g., logistic regression) or non-convex (e.g., Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM)). The training steps of HFL can be summarized as follows:

1) At the beginning of round t, each ES m randomly selects a subset of clients $\mathbf{s}_m^t \subseteq \mathcal{N}_m^t$ in its coverage area. Even if a client is in the overlapping area, it is only allowed to communicate with one ES in one round. We assume that the HFL network contains a backhaul link to transmit the selected clients to avoid some clients being selected on multiple ESs. Each client n selected by ES m downloads the edge model $\mathbf{w}_m^t$ and sets it to be the local model $\mathbf{w}_n^t = \mathbf{w}_m^t$, $\forall n \in \mathbf{s}_m^t$.

2) Then, each client n takes E epochs to update its own local model by Stochastic Gradient Descent (SGD) from its dataset $\mathcal{D}_n$ as follows:$\mathbf{w}_n^{t+e+1} = \mathbf{w}_n^{t+e} - \eta_t g(\mathbf{w}_n^{t+e}; \xi_n^{t+e})$, where $e = 0, 1, \dots, E-1$, η_t is the learning rate, and $g(\mathbf{w}_n^{t+e}; \xi_n^{t+e})$ is the stochastic gradient of $F_n(\mathbf{w})$ (i.e., $\mathbb{E}_{\xi_n^{t+e} \sim \mathcal{D}_n}[g(\mathbf{w}_n^{t+e}; \xi_n^{t+e})] = \nabla F_n(\mathbf{w}_n^{t+e})$).

3) After E local training epochs, client $n \in \mathcal{S}_m^t$ uploads the local model updates $\Delta_n^t \triangleq \mathbf{w}_n^{t+E-1} - \mathbf{w}_n^t$ to the ES m. Instead of aggregating all local models on CS at the end of round t [107, 80, 62] of conventional FL, local model updates are averaged within ES m to be edge model $\mathbf{w}_m^{t+1}$, called edge aggregation, which is given as follows:

$$\mathbf{w}_m^{t+1} = \mathbf{w}_m^t + \frac{1}{S_m^t} \sum\nolimits_{n \in \mathbf{s}_m^t} \Delta_n^t, \tag{3.2}$$

4) After T_{ES} rounds of edge aggregation, global model $\mathbf{w}^t$ is computed by $\mathbf{w}^t = \frac{1}{M} \sum_{m=1}^{M} \mathbf{w}_m^t$, $\forall t = \{T_{\mathrm{ES}}, 2T_{\mathrm{ES}}, \dots\}$ from all M ESs, called global aggregation. Then, each ES m downloads the global model to be its edge model $\mathbf{w}_m^t = \mathbf{w}^t$.

Repeating the above four steps with a sufficiently large round T, NO will achieve the global model $\mathbf{w} = \mathbf{w}^T$ and stop the training process. HFL has been demonstrated that it achieves a linear speedup of convergence to conventional FL algorithms [93, 159].

3.2.2 Cost of Client Selection

Since clients usually do not belong to NO, clients are required to charge NO for the number of requested computation resources for collecting datasets and processing the local training to achieve the learning goal. At the beginning of each edge aggregation round, each client reveals its available computation resources y_n^t to NO, which includes CPU frequency, RAM and storage, etc., to process the current local training updates.

Each client sets a price for its computation resources. Let $c_n(y_n^t)$ denote the price charged by client n, where $c_n(\cdot)$ is a non-decreasing mapping function related to the price for computation resources y_n^t. Due to the limited rental budget $\tilde{B}$ of NO, for any edge aggregation round t, the client selection decision NO must satisfy the budget constraint $\sum_{m\in\mathcal{M}}\sum_{n\in\mathbf{s}_m^t} c_n(y_n^t) \leq \tilde{B}$.

3.2.3 Deadline Based HFL

In summary, an edge aggregation consists of four stages: Download Transmission (DT), Local Computation (LC), Upload Transmission (UT) and Edge Computation (EC).

In DT stage, the selected client $n \in \mathbf{s}_m^t$ downloads the current edge model from the ES m. Followed by Shannon's equation, the channel state of DT $c_{\mathrm{DT},n}^t$ is calculated by:

$$c_{\mathrm{DT},n}^t = \log_2(1 + P_n^t g_{\mathrm{DT},n}^t / N_0), \tag{3.3}$$

where P_n^t is the transmission power, $g_{\mathrm{DT},n}^t$ is the downlink wireless channel gain and N_0 is the noise power. Let a_{DT} denote the downloading data size (i.e., size of edge model $\mathbf{w}_m^t$) and the allocated bandwidth is b_n^t in the edge aggregation round t. Therefore, thus the DT time for client n is $\tau_{\mathrm{DT},n}^t = a_{\mathrm{DT}}/(b_n^t c_{\mathrm{DT},n}^t)$.

Once the client n receives $\mathbf{w}_m^t$, training comes to the LC stage (i.e., it updates the local model using its own dataset $\mathcal{D}_n$ according to Eq. (2.2)). The LC time of each client is determined by the local computation resources y_n^t in the current round t. Given the com-

Figure 3.2: MNIST under logistic regression and CIFAR-10 under CNN.

putation resources $y_n^t > 0$, the LC time can be obtained as $\tau_{\mathrm{LC},n}^t(y_n^t) = q/y_n^t$, where q is the computation workload, which is based on the complexity of learning model and data.

When the LC is finished, client n uploads its local model updates Δ_n^{t+1} to the ES m. Similar to the channel state definition of DT in Eq. (3.3), the channel state of UT is $c_{\mathrm{UT},n}^t = \log_2(1+P_n^t g_{\mathrm{UT},n}^t/N_0)$ and UT time is $\tau_{\mathrm{UT},n}^t = a_{\mathrm{UT}}/(b_n^t c_{\mathrm{UT},n}^t)$, where $g_{\mathrm{UT},n}^t$ is the uplink channel gain and a_{UT} is the uploading data size (i.e., size of Δ_n^{t+1}).

Finally, if the local model updates of all selected clients are received by ESs, the edge models should be computed in Eq. (3.2). The EC time is $\tau_{\mathrm{EC},m}^t = q_m/y_m$, where q_m is the edge model workload and y^m is the process capacity of the ES m. Note that $\tau_{\mathrm{EC},m}$ should be different of every ES m. However, since the EC stage only takes the average calculation of the received local model updates according to Eq. (3.2) and the capacity of ES q_m is always very large compared to clients, this does not waste much running time compared to the other three stages. Therefore, the difference of $\tau_{\mathrm{EC},m}$ of all ESs is too small to be negligible, and hence NO does not need to consider the influence of $\tau_{\mathrm{EC},m}$. As such, the training time of client n is defined as follows:

$$\tau_n^t(y_n^t) = \tau_{\mathrm{DT},n}^t + \tau_{\mathrm{LC},n}^t(y_n^t) + \tau_{\mathrm{UT},n}^t = \frac{a_{\mathrm{DT},n}^t}{b_n^t c_{\mathrm{DT},n}^t} + \frac{q}{y_n^t} + \frac{a_{\mathrm{UT},n}^t}{b_n^t c_{\mathrm{UT},n}^t}, \quad \forall n, t. \tag{3.4}$$

Due to some physical limitations (e.g., low computation capability and unstable communication), some clients may incur huge training latency in one edge aggregation round. Therefore, the deadline-based FL [113, 152, 178] is more realistic to deal with straggler

clients. Specifically, ESs drop the clients whose the local model updates cannot be received before the deadline $\tau_{\text{dead},m}$ (i.e., client n such that $\tau_n^t > \tau_{\text{dead},m}$). In this chapter, we consider deadline-based HFL. Therefore, the edge aggregation can be reformulated:

$$\mathbf{w}_m^t = \begin{cases} \frac{1}{\sum_{n\in\mathbf{s}_m^t} X_n^t} \sum_{n\in\mathbf{s}_m^t} X_n^t \mathbf{w}_n^t, & \text{if } \sum_{n\in\mathbf{s}_m^t} X_n^t \geq Z \\ \frac{1}{Z} \sum_{\tau_n^t \leq \tau^Z} \mathbf{w}_n^t, & \text{else} \end{cases} \tag{3.5}$$

where X_n^t is a binary random variable representing whether client n's model update can be received before the deadline (i.e., if $\tau_n^t \leq \tau_{\text{dead},m}$, $X_n^t = 1$; otherwise, $X_n^t = 0$), and τ^Z is the training time of the Z-th fastest client. In Figure 3.2, we can see that if each ES only receives 3 local model updates, the training performance has large degradation and variance. In order to guarantee a minimum level of training performance, we require that at least Z local model updates must be received for edge aggregation. Therefore, in case less than Z clients' updates are received before the deadline, the system has to wait for some additional time $\tau^Z - \tau_{\text{dead},m}$. For practical values of $\tau^Z - \tau_{\text{dead},m}$, the probability of having less than Z client updates received before the deadline is small. For analysis convenience, we assume that at least Z client updates can be received before the deadline in every edge aggregation round. In addition, we assume that the deadline of all ESs are set the same, $\tau_{\text{dead},m} = \tau_{\text{dead}}, \forall m$. The extension to heterogeneous deadlines is straightforward.

3.2.4 Utility Function of Client Selection of HFL

Some existing HFL studies [159, 93, 97] have demonstrated that the convergence speed depends on the number of participating clients in each edge aggregation round for both strongly convex and non-convex HFL (i.e., the more clients participated, the faster convergence speed). In order to support the theoretical results, we show the training performance on our simulated HFL network with $M = 3$ and $N = 50$, and it is observed that more participating clients on ESs can improve the performance in both the strongly convex and non-convex HFL settings.

For now, we consider strongly convex HFL, where the convergence speed is linearly dependent on the number of participating clients. The client selection policy for non-convex HFL will be developed in Section 3.4. As in [113, 152, 178], not all the selected clients in $\mathbf{s}_m^t$ may reach the EC stage (i.e., $\sum_{n\in\mathbf{s}_m^t} X_n^t \leq S_m^t$) due to straggler drop-out. To achieve targeted convergence criteria, NO thus needs to run more FL rounds, thereby incurring a higher training cost. Therefore, it is necessary to develop an efficient client selection policy to improve the convergence speed for HFL, where more clients can participate in every round without dropping out. Let $\mathbf{X}_m^t = \{X_{n,m}^t\}_{\forall n\in\mathcal{N}_m^t}$, then the utility of the client selection decision on ES m is defined as:

$$\mu(\mathbf{s}_m^t; \mathbf{X}_m^t) = \sum\nolimits_{n\in\mathbf{s}_m^t} X_{n,m}^t, \tag{3.6}$$

Further, let $\mathbf{s}^t = \{\mathbf{s}_1^t, \mathbf{s}_2^t, \dots, \mathbf{s}_M^t\}$ denote the client selection decision of the overall system and $\mathbf{X}^t = \{\mathbf{X}_1^t, \mathbf{X}_2^t, \dots, \mathbf{X}_M^t\}$. Therefore, the utility function of the whole HFL network is defined as:$\mu(\mathbf{s}^t; \mathbf{X}^t) = \frac{1}{M}\sum_{n\in\mathcal{M}}\sum_{n\in\mathbf{s}_m^t} X_{n,m}^t$.

3.2.5 Client Selection Problem Formulation

The client selection problem for NO is a sequential decision-making problem. The goal of NO is to make a selection decision $\mathbf{s}^t, \forall t$ to maximize the cumulative utility for a total of T aggregation rounds. If an ES selects very few clients, its training performance may be degraded and the computation resources of ESs may be wasted. Since $\mathcal{N}_m^t$ is a time-variant in each edge aggregation due to the client movement, we assume that the location of client n is uniformly distributed in the HFL network. Moreover, as T_{ES} is usually set larger than 1, NO cannot allocate the total budget $\tilde{B}$ to M ESs in each edge aggregation round. Therefore, we consider that NO equally divides the budget among the ESs (i.e., for each m, its budget is $B = \tilde{B}/M$) from the expectation perspective. Assuming that NO knows a priori whether a selected client can return its model updates to the corresponding edge server in time, namely

$\mathbf{X}^t, \forall t$, then the client selection problem is formulated:

$$\text{P1:} \quad \max_{\{\mathbf{s}^t\}_{t=1}^T} \quad \sum_{t=1}^{T} \mu(\mathbf{s}^t; \hat{\mathbf{X}}^t) \tag{3.7a}$$

$$\text{s.t.} \quad \sum_{n \in \mathbf{s}_m^t} c_n(y_n^t) \leq B, \quad \forall m \in \mathcal{M} \tag{3.7b}$$

$$\mathbf{s}_m^t \subseteq \mathcal{N}_m^t, \quad \forall m, t \tag{3.7c}$$

$$\mathbf{s}_m^t \cap \mathbf{s}_{m'}^t = \emptyset, \quad m, m' \in \mathcal{M}, \forall t. \tag{3.7d}$$

The following challenges should be addressed to solve the client selection problem in HFL networks: (i) For maximizing the expected training utility of HFL, it is necessary to precisely estimate the selected clients in each edge aggregation round. In addition, since NO does not have enough experience to determine the selected clients at the first several rounds (i.e., cold start), collecting the historical data for estimation is important for this policy. (ii) With the successful participated clients estimation, how to optimize the selection decision on each ES under the limited budget should be carefully considered, because the high variance of the number of participated clients on each ES degrades the training performance. Therefore, we equally separate the total budget for each ES (constraint (3.7b)). (iii) Due to the movement of each client and the overlapping areas across all ESs, the available connecting ESs can be considered as time-varying (constraint (3.7c)), we must decide the client-ES pairs, especially for the clients in overlapping areas, which brings more difficulties to make an efficient client selection decision. Note that constraint (3.7d) can guarantee that each client only can be selected to communicate at most one ES. (iv) Since the selection decisions are based on the estimated participated clients $\hat{\mathbf{X}}^t$, the accuracy of participated clients estimation will directly influence the training utility of NO. After finishing the edge model updates, ESs can process the client selection for the next round (i.e., processing in DT, LC, and UT stages). Furthermore, since the large computational capacity y_m^t is large enough, it is reasonable to assume that ESs can finish the decision before receiving the new local model updates. Therefore, making a client selection policy does not impact the training time. The following

section will propose a policy based on the Multi-Armed Bandits (MAB) to address the mentioned challenges.

3.3 Context-Aware Online Client Selection for Strongly Convex HFL

In this section, we formulate our client selection problem of HFL as a Contextual Combinatorial Multi-Armed Bandit (CC-MAB). The combinatorial property is because NO pays computation resources from multiple clients for maximizing the training utility. The contextual property is because NO leverages contexts associated with clients to infer their participated probabilities. In this chapter, whether successfully participating in the corresponding ES depends on many side factors, which are collaboratively referred to as context. We use contextual information to help infer the number of participating clients.

In CC-MAB, NO observes the context of clients at the beginning of each edge aggregation round before making the client selection decision. Recall that the participated probability of a client-ES pair depends on $r^t_{\mathrm{DT},n}, y^t_n$ and $r^t_{\mathrm{UT},n}$ in Eq. (3.4). At each edge aggregation round t, ES can measure the the channel state $c^t_{\mathrm{DT},n}$ by inferring the received signal strength of the received local model updates [8, 30]. Based on $c^t_{\mathrm{DT},n}$ and bandwidth b^t_n, ESs can compute the DT rate $r^t_{\mathrm{DT},n}$. In addition, since the movement speed of clients is much slower than the transmission speed of wireless signals, while NO cannot know the UT rate $r^t_{\mathrm{UT},n}$, it is not difficult to be inferred by $r^t_{\mathrm{DT},n}$ (suppose that clients do not locate in the same area in each edge aggregation round). Therefore, we set $c^t_{\mathrm{DT},n}$ as context and use this information to help significantly improve the participated probabilities of client-ES pairs.

Let $\phi^t_{n,m} \in \Phi$ denote the context of client-ES pair (n, m) in edge aggregation round t. Without loss of generality, we normalize $\phi^t_{n,m}$ in a bounded space $\Phi = [0, 1]^2$ using min-max feature scaling. Let $\boldsymbol{\phi} = \{\phi^t_{n,m}\}_{\forall m, n \in \mathcal{N}^t_m}$ denote the context of all clients on ESs. The context of all clients on ESs are collected in $\boldsymbol{\phi}^t = \{\phi^t_{n,m}\}_{\forall m, n \in \mathcal{N}^t_m}$. Whether successful participation of client n on ES m is a random variable parameterized by the context $\phi^t_{n,m}$. We slightly simplify the notation of selected clients and define the context-aware $X^t_{n,m}(\phi^t_{n,m})$.

Specifically, $X_{n,m}$ is a mapping function for each client-ES pair (n, m), since the training time of clients is usually location-dependent (e.g., the distance between the client and ES, the communication environment, and other processing tasks on a client). We further define $p_{n,m}(\phi^t_{n,m}) \triangleq \mathbb{E}[X_{n,m}(\phi^t_{n,m})]$ as the expected value (i.e., the participated probability $X^t_{n,m} \sim \text{Bernoulli}(p^t_{n,m})$) of $X_{n,m}(\phi^t_{n,m})$.

3.3.1 Oracle Solution and Regret

Similar to the existing CC-MAB studies [29, 28], before providing our policy design, we first give an Oracle benchmark solution to the client selection problem of HFL by assuming that the NO knows the context-aware successful participated probability $p^t_{n,m}(\phi^t_{n,m})$, $\forall m, n \in \mathcal{N}^t_m$. In this ideal setting, the utility function $\mu(\mathbf{s}^t; \mathbf{p}^t)$ is perfectly known by NO, and thus we can get the optimal value of the client selection problem. The long-term selection problem P1 can be decomposed into T independent subproblems in each edge aggregation round:

$$\text{P2:} \quad \max_{\mathbf{s}^t} \quad \mu(\mathbf{s}^t; \mathbf{p}^t) \tag{3.8a}$$

$$\text{s.t.} \quad \sum\nolimits_{n \in \mathbf{s}^t_m} c_n(y^t_n) \leq B, \quad \forall m \in \mathcal{M} \tag{3.8b}$$

$$\mathbf{s}^t_m \subseteq \mathcal{N}^t_m, \quad \forall m, t \tag{3.8c}$$

$$\mathbf{s}^t_m \cap \mathbf{s}^t_{m'} = \emptyset, \quad m, m' \in \mathcal{M}, \forall t. \tag{3.8d}$$

P2 is a combinatorial optimization problem with M Knapsack and a Matroid constraints. The combinatorial property is because NO should choose a proper client selection decision to optimize participated probabilities on all ESs in order to achieve higher convergence speed. Knapsack constraints are from the constraint (3.8b), which bounds computation resources payment on each ES.

To prove that (3.8c) is a matroid constraint, we first state the definition of the matroid. A matroid $\mathcal{E} = ((X), (I))$ is a system with independents sets, in which $\mathcal{X}$ is a finite set (named the ground set) and $\mathcal{I}$ represents the set of independent subsets of $\mathcal{X}$. It has the three following properties: (1) $\emptyset \in \mathcal{I}$ and $\mathcal{X}$ has at least one subset of $\mathcal{X}$; (2) For each

$A \subset B \subset \mathcal{X}$, if $A \in \mathcal{I}$, then $B \in \mathcal{I}$; (3) If $A, B \in \mathcal{I}$, and $|A| < |B|$, then $\exists \in B \setminus A$ such that $A\{x\} \in \mathcal{I}$.

In the subproblem P2, let $\mathcal{X} = \cup_{m \in \mathcal{M}} \mathcal{N}_m^t$ denote the ground set of matroid $\mathcal{E} = (\mathcal{X}, \mathcal{I})$, and $\mathcal{I} = \{I_1, I_2, ...\}$ consists of subsets of $\mathcal{X}$ (i.e., $I_1 \subseteq \mathcal{X}$, $I_2 \subseteq \mathcal{X}$, ...), where all $I \in \mathcal{I}$ includes at most one client from $\mathcal{N}_m^t$ for each $m \in \mathcal{M}$. We can write I as $I = \cup_{m \in \mathcal{M}} \mathbf{s}_m^t$, s.t. $\mathbf{s}_m^t \in \mathcal{N}_m^t$, $\forall m$. In this chapter, $\mathcal{I}$ is the set of all feasible client selection decisions. Therefore, it can be verified that Eq. (3.8c) is a matroid constraint [28].

Based on our analysis, it is easy to observe that P2 is NP-hard, and hence it can be solved by brute force, if the size of the HFL network is moderate. If the HFL network is too large, NO can use some commercial software to obtain the optimal solution (e.g., CPLEX [32]). For simplicity, we define the optimal Oracle solution for each P2 in edge aggregation round t is $\mathbf{s}^{\text{opt},t}$. However, in practice, the prior knowledge of participated clients is infeasible, and thus the NO has to make a selection decision $\mathbf{s}^t$ based on the estimated participated clients $\hat{\mathbf{X}}^t$ in each edge aggregation round. Intuitively, NO should design an online client selection policy to choose $\mathbf{s}^t$ based on the estimation $\hat{\mathbf{X}}^t$. The performance of an online client selection policy is calculated by utility loss compared with the Oracle solution, called regret. Suppose that we have a selection sequence $\{\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^T\}$ given by a policy, the expected regret is:

$$\mathbb{E}[R(T)] = \sum_{t=1}^{T} (\mathbb{E}[\mu(\mathbf{s}^{\text{opt},t}; \mathbf{X}^t)] - \mathbb{E}[\mu(\mathbf{s}^t; \mathbf{X}^t)]). \tag{3.9}$$

The expectation is concerning with respect to the decisions made by the client selection decision policy and the participated clients over contexts.

3.3.2 Context-aware Online Client Selection Policy

Now, we will present our online client selection decision policy name Context-aware Online Client Selection (COCS). The COCS policy is designed based on CC-MAB. In edge aggregation round t, the process of COCS of NO is operated sequentially as follows: (i) NO observes the contexts of all client-ES pairs $\boldsymbol{\phi}^t = \{\phi_{n,m}^t\}_{n,m \in \mathcal{N}_m^t}$, $\phi_{n,m}^t \in \Phi$. (ii) NO determines

its selection decision $\mathbf{s}^t$ based on the observed context information $\boldsymbol{\phi}^t$ in the current round t and the knowledge learned from the previous $t-1$ rounds. (iii) The selection decision $\mathbf{s}^t$ is applied. If $s_n^t \neq 0$, $\forall n \in \mathbf{s}_m^t$, the clients located in the coverage area of ES m can be selected by ES m for training in round t. (iv) At the end of each edge aggregation round, the local model updates Δ_n^t from which clients are observed by all ESs, which is then used to update the estimated participated clients $\hat{X}_{n,m}(\phi_{n,m}^t)$ from the observed context $\phi_{n,m}^t$ of client-ES pair (n,m).

It has two parameters $K(t)$ and h_T to be designed, where $K(t)$ is a deterministic and monotonically increasing function used to identify the under-explored context, and h_T decides how we partition the context space. The COCS policy is stated as follows:

For the initialization phase, given parameter h_T, the proposed policy first creates a partition denoted by $\mathcal{L}_T$ for the context space $\Phi = [0,1]^2$, which splits Φ into $(h_T)^2$ sets. Each set is a 2-dimensional hypercube with size $\frac{1}{h_T} \times \cdots \times \frac{1}{h_T}$. Note that h_T is an important input parameter to guarantee policy performance. For each hypercube $l \in \mathcal{L}_T$, the NO keeps a counter $C_{n,m}^t(l)$ for each client $n \in \mathcal{N}$ and each ES $m \in \mathcal{M}$. For the tuple (n,m,l) of a counter $C_{n,m}^t(l)$ for each client-ES pair (n,m), we define a selection event $V_{n,m,l}$ that represents a selection decision satisfying the three following conditions: 1) the client $n \in \mathcal{N}_m^t$ is selected to an ES m; 2) the ES m successfully receives the client n before τ_{dead} (i.e., $\tau_n^t \leq \tau_{\text{dead}}$); 3) the context of client-ES pair (n,m) belongs to l (i.e., $\phi_{n,m}^t \in l$). The counter $C_{n,m}^t(l)$ stores the number of times that the event $V_{n,m,l}$ occurs until edge aggregation round t. Each ES m also saves an experience $\mathcal{E}_{n,m}^t(l)$ for each client n and each hypercube l, which contains the observed participated clients indicators when a selection event $V_{n,m,l}$ occurs. The experience $\mathcal{E}_{n,m}^t(l)$ is useful for making the future decision of whether to come into the exploration or exploitation phase. Based on the observed participation indicators in $\mathcal{E}_{n,m}^t(l)$, the estimated participated probability for a selection event $V_{n,m,l}$ is computed by: $\hat{p}_{n,m}^t(l) = \frac{1}{C_{n,m}^t(l)} \sum_{X \in \mathcal{E}_{n,m}^t(l)} X$. In each edge aggregation round t, the COCS policy has the following phases:

For the hypercube identification phase, if the local model updates Δ_n^t of client $n \in \mathcal{N}_m^t$ can be successfully received by an ES m in edge aggregation round t, we obtain that $l_{n,m}^t$ is the hypercube for the context $\phi_{n,m}^t$, the estimated participated probability of client n on ES m is $\hat{X}_{n,m}^t = \hat{p}_{n,m}^t(l_{n,m}^t)$. Let $\hat{\mathbf{X}}^t = \{\hat{X}_{n,m}^t\}_{\forall m, n \in \mathcal{N}_m^t}$ denote the collection of all the estimated participated probabilities. For making a client selection decision, the COCS policy needs to check whether these hypercubes have been explored sufficiently to ensure enough accuracy of the estimated participated probability for each client-ES pair (n, m). Therefore, we define under-explored hypercubes $\mathcal{L}_m^{\mathrm{ue}}(\boldsymbol{\phi}^t)$ for the ES m in edge aggregation round t as follows:

$$\mathcal{L}_m^{\mathrm{ue},t} \triangleq \left\{ l \in \mathcal{L}_T \middle| \begin{array}{l} \exists \phi_{n,m}^t \in \boldsymbol{\phi}^t, \phi_{n,m}^t \in l, \tau_n^t \leq \tau_{\mathrm{dead}} \\ \text{and } C_{n,m}^t(l) \leq K(t) \end{array} \right\}. \tag{3.10}$$

Also, let $\mathcal{N}_m^{\mathrm{ue},t}(\boldsymbol{\phi}^t) \triangleq \{n \in \mathcal{N}_m^t | l_{n,m}^t \in \mathcal{L}_m^{\mathrm{ue},t}(\boldsymbol{\phi}^t)\}$ denote the collection of the under-explored client n for each ES m. The challenge of COCS policy is how to decide whether the current estimated participated clients are accurate enough to guide the client selection decision in each edge aggregation round, which is referred to as exploitation or more training results, which is referred to as exploration. COCS policy aims to balance the exploration and exploitation phases to maximize the utility of NO up to a finite round T. Now, we come into the exploration phase. Firstly, let $\mathcal{N}_m^{\mathrm{ue},t}(\boldsymbol{\phi}^t) \triangleq \{n \in \mathcal{N}_m^t | \mathcal{N}_m^{\mathrm{ue},t} \neq \emptyset\}$ denote an ES m has under-explored clients, and $\mathcal{N}_m^{\mathrm{ed},t}(\boldsymbol{\phi}^t) \triangleq \mathcal{N} \setminus \mathcal{N}^{\mathrm{ue},t}(\boldsymbol{\phi}^t)$ denote the ES m does not have under-explored clients. If the ES m has a non-empty $\mathcal{N}_m^{\mathrm{ue},t}$, then COCS enters the exploration phase, which includes the following cases:

1) All the clients have under-explored ESs. Intuitively, NO hopes to receive more local training updates Δ_n^t. Therefore, COCS policy aims to select as many clients that have under-explored ESs sequentially solved by the following optimization:

$$\max_{\mathbf{s}^t} \ |\mathbf{s}^t| \quad \text{s.t. (3.8b), (3.8c), (3.8d),} \tag{3.11}$$

where $|\mathbf{s}^t|$ is the size of the collection $\mathbf{s}^t = \{\mathbf{s}_1^t, \mathbf{s}_2^t, \ldots, \mathbf{s}_M^t\}$.

2) Part of ESs have under-explored clients $\exists \mathcal{N}_m^{\mathrm{ue},t} \neq \emptyset$. We divide this case into two stages: NO firstly selects ESs that have under-explored clients $m \in \mathcal{N}_m^{\mathrm{ue},t}$ by solving the following optimization:

$$\max_{\tilde{\mathbf{s}}^t} \quad |\tilde{\mathbf{s}}^t| \tag{3.12a}$$

$$\text{s.t.} \quad \sum\nolimits_{n \in \tilde{\mathbf{s}}_m^t} c_n(y_n^t) \leq B, \quad \forall n \in \mathcal{N}_m^{\mathrm{ue},t}, \forall m \in \mathcal{M} \tag{3.12b}$$

$$s_m^t \in \mathcal{N}_m^t \cup \{null\}, \quad \forall n \in \mathcal{N}_m^{\mathrm{ue},t} \tag{3.12c}$$

$$\tilde{\mathbf{s}}_m^t \cap \tilde{\mathbf{s}}_{m'}^t = \emptyset, \quad m, m' \in \mathcal{M}, \forall t. \tag{3.12d}$$

where $\tilde{\mathbf{s}}^t$ is client selection decision on ES m that has under-explored clients and $|\tilde{\mathbf{s}}^t|$ is the size of the collection $\tilde{\mathbf{s}}^t = \{\tilde{\mathbf{s}}_1^t, \tilde{\mathbf{s}}_2^t, \dots, \tilde{\mathbf{s}}_M^t\}$. Secondly, ESs aim to select the explored clients $\forall n \in \mathcal{N}_m^{\mathrm{ed},t}$. Here, we assume that there exists ESs that $B - \sum_{n \in \tilde{\mathbf{s}}_m^t} c_n(y_n^t) \geq c^{\min,t}, m \in \mathcal{M}$, where $c^{\min,t} = \min_{n \in \mathcal{N}_m^{\mathrm{ed},t}} c_n(y_n^t), \forall m$. Therefore, ESs can select the clients $n \in \mathcal{N}_m^{\mathrm{ed},t}$ with the following constraint:

$$\sum\nolimits_{n \in \mathbf{s}_m^t \setminus \tilde{\mathbf{s}}_m^t} c_n(y_n^t) \leq B - \sum\nolimits_{n \in \tilde{\mathbf{s}}_m^t} c_n(y_n^t), \ \forall n \in \mathcal{N}_m^{\mathrm{ed},t}. \tag{3.13}$$

If not, NO does not need to select clients in $\mathcal{N}_m^{\mathrm{ed},t}$ due to no budget left. Under this condition, the client selection decisions are jointly optimized for the following optimization:

$$\max_{\mathbf{s}^t} \ \mu(\mathbf{s}^t; \hat{\mathbf{X}}^t) \tag{3.14a}$$

$$\text{s.t. } B - \sum\nolimits_{n \in \tilde{\mathbf{s}}_m^t} c_n(y_n^t) \geq c^{\min,t}, \quad m \in \mathcal{M}, \tag{3.14b}$$

$$(3.12\text{b}), (3.12\text{c}), (3.13). \tag{3.14c}$$

For the exploitation phase, if the set of under-explored clients is empty (i.e., $\mathcal{N}_m^{\mathrm{ue},t} = \emptyset, \forall m$), then COCS policy enters the exploitation phase. The optimal client selection decision $\mathbf{s}^t$ is derived by solving P2 from the current estimated participated clients $\hat{\mathbf{X}}^t$:

$$\max_{\mathbf{s}^t} \ \mu(\mathbf{s}^t; \hat{\mathbf{X}}^t) \quad \text{s.t. } (3.8\text{b}), (3.8\text{c}). \tag{3.15}$$

To this end, it comes into the update phase. After selecting the client-ES pair, COCS policy observes whether the local model updates of selected clients can be received before the deadline τ^{dead}; then, it updates $\hat{p}^t_{n,m}(I)$ and $C^t_{n,m}(I)$ of each hypercube $I \in \mathcal{L}_T$.

3.3.3 Performance Analysis

To present an upper performance bound of the COCS policy, we make the following assumption that the participating clients are similar when their contexts are similar. It is formalized by the following Hölder condition [29, 28], which is defined as follows:

Assumption 5. *(Hölder Condition). If a real function f on D-dimensional Euclidean space satisfies Hödel condition, there exists $L > 0$, $\alpha > 0$ such that for any $\phi, \phi' \in \Phi$, it holds that $|f_n(\phi) - f_n(\phi')| \leq L\|\phi - \phi'\|^\alpha$ for an arbitrary client $n \in \mathcal{N}$, where $\|\cdot\|$ is the Euclidean norm.*

By providing the design of the input parameters $K(t)$ and h_T, we show that COCS policy achieves a sublinear $R(T) = O(T^\gamma)$ with $\gamma < 1$, which guarantees that COCS has an asymptotically optimal performance. This means that the online client selection decision via COCS policy converges with the Oracle solution. Because any edge aggregation round is either in the exploration or exploitation phase, the regret can be divided into two parts $R(T) = R_{\text{explore}}(T) + R_{\text{exploit}}(T)$, where $R_{\text{explore}}(T)$ and $R_{\text{exploit}}(T)$ are the regrets due to exploration and exploitation phases, respectively. The total regret bound is achieved by separately bounding these two parts. Therefore, we present the following two lemmas for bounding exploration and exploitation regrets.

Lemma 2. *(Bound of $\mathbb{E}[R_{explore}(T)]$.) Given the input parameters $K(t) = t^z \log(t)$ and $h_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{2}$, the regret $\mathbb{E}[E_{explore}(T)]$ is bounded by:*

$$\mathbb{E}[E_{explore}(T)] \leq \frac{4N^2MB}{c^{min}}(T^{z+2\gamma}\log(T) + T^{2\gamma}),$$

where $c^{min} = \min_{y^t_n, \forall n,t} c_n(y^t_n)$.

Proof. See in online Appendix A, available in the online supplemental material [127]. □

Lemma 2 shows that the order of $R_{\text{explore}}(T)$ is determined by the control function $K(T)$ and the number of hypercubes $(h_T)^D$ in partition $\mathcal{L}_T$.

Lemma 3. *(Bound of $\mathbb{E}[R_{exploit}(T)]$.) Given $K(t) = t^z \log(t)$ and $h_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{2}$, if the Hölder condition holds true and the additional condition $2H(t) + \frac{2NMB}{c^{min}} L 2^{\frac{\alpha}{2}} h_T^\alpha \leq A t^\theta$ is satisfied with $H(t) > \frac{NMB}{c^{min}} t^{-\frac{z}{2}}$, $A > 0$, $\theta < 0$, for all t, then $\mathbb{E}[R_{exploit}(T)]$ is bounded by:*

$$\mathbb{E}[R_{exploit}(T)] \leq \frac{NMB}{c^{min}} \Bigg(\sum_{k=1}^{B/c^{min}} \binom{N}{k} \Bigg) \frac{\pi^2}{3} + \frac{3NMB}{c^{min}} L 2^{\frac{\alpha}{2}} T^{1-\gamma\alpha} + \frac{A}{1+\theta} T^{1+\theta},$$

where $c^{min} = \min_{y_n^t, \forall n,t} c_n(y_n^t)$.

Proof. See in online Appendix B, available in the online supplemental material [127]. □

Lemma 3 indicates that the regret of exploitation $\mathbb{E}[R_{\text{exploitation}}(T)]$ depends on the choice of z and γ with an additional condition being satisfied. Based on the above two Lemmas, we will have the following Theorem for the upper bound of the regret $\mathbb{E}[R(T)]$.

Theorem 4. *(Bound of $\mathbb{E}[R(T)]$.) Given the input parameters $K(t) = t^z \log(t)$ and $h_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{2}$, if the Hölder condition holds true and the additional condition $2H(t) + \frac{2NMB}{c^{min}} L 2^{\frac{\alpha}{2}} h_T^\alpha \leq A t^\theta$ is satisfied with $H(t) > \frac{NMB}{c^{min}} t^{-\frac{z}{2}}$, $A > 0$, $\theta < 0$, for all t, then the regret $\mathbb{E}[R(T)]$ can be bounded by:*

$$\begin{aligned}\mathbb{E}[R(T)] \leq & \frac{4N^2MB}{c^{min}} (T^{z+2\gamma} \log(T) + T^{2\gamma}) \\ & + \frac{NMB}{c^{min}} \Bigg(\sum_{k=1}^{B/c^{min}} \binom{N}{k} \Bigg) \frac{\pi^2}{3} + \frac{3NMB}{c^{min}} L 2^{\frac{\alpha}{2}} T^{1-\gamma\alpha} + \frac{A}{1+\theta} T^{1+\theta},\end{aligned}$$

where $c^{min} = \min_{y_n^t, \forall n,t} c_n(y_n^t)$.

Proof. See in online Appendix C, available in the online supplemental material [127]. □

The regret upper bound in Theorem 4 is given by properly choosing input parameters $K(t)$ and h_t. However, the values of z, γ, A and θ are not deterministic. Next, we will show the regret upper bound of $\mathbb{E}[R(T)]$ in these parameters design.

Theorem 5. *(Regret upper bound). If we select* $z = \frac{2\alpha}{3\alpha+2} \in (0,1)$, $\gamma = \frac{z}{2\alpha}$, $\theta = -\frac{z}{2}$, $A = \frac{2NMB}{c^{min}} t^{-\frac{z}{2}} + \frac{2NMB}{c^{min}} L 2^{\frac{\alpha}{2}}$, *and COCS algorithm runs with these parameters, the regret* $\mathbb{E}[R(T)]$ *can be bounded by:*

$$\mathbb{E}[R(T)] \leq \frac{4N^2MB}{c^{min}}(\log(T) T^{\frac{2\alpha+2}{3\alpha+2}} + T^{\frac{2}{3\alpha+2}}) + \frac{NMB}{c^{min}}\left(\sum_{k=1}^{B/c^{min}} \binom{N}{k}\right)\frac{\pi^2}{3} + \left(3L2^{\frac{\alpha}{2}} + \frac{2 + L2^{\frac{\alpha}{2}}}{(2\alpha+2)/(3\alpha+2)}\right)\frac{NMB}{c^{min}} T^{\frac{2\alpha+2}{3\alpha+2}},$$

where $c^{min} = \min_{y_n^t, \forall n,t} c_n(y_n^t)$.

The dominant order of the regret $\mathbb{E}[R(T)]$ *is* $O\left(\frac{4N^2MB}{c^{min}} T^{\frac{2\alpha+2}{3\alpha+2}} \log(T)\right)$.

Proof. See in online Appendix C, available in the online supplemental material [127]. □

The dominant order of regret upper bound, which indicates the COCS policy in Theorem 5 is sublinear. In addition, the regret bound is valid for any total rounds T, and it can be used to characterize the convergence speed of HFL.

3.3.4 Complexity Analysis

The space complexity of COCS policy is determined by the number of counters $C_{n,m}^t(l)$ and experiences $\mathcal{E}_{n,m}^t(l)$ maintained for hypercubes. Because the counter is an integer for each hypercube, the space complexity is determined by the number of hypercubes. The experience $\mathcal{E}_{n,m}^t(l)$ is a set of observed successfully participating clients' records up to round t, which requires a higher memory. However, it is unnecessary to store all historical records, since most estimators can be updated recursively. Therefore, the NO only needs to keep the current participated clients' estimation for a hypercube. If COCS is run with the parameters in Theorem 5, the number of hypercubes is $(h_T)^2 = \lceil T^{\frac{1}{3\alpha+2}} \rceil^2$, and thus the required space is sublinear in total rounds T. This means that when $T \to \infty$, COCS will require infinite memory. In the practical implementations, NO only needs to keep the counters and experiences of hypercubes to which at least one of the observed contexts occurs. Therefore,

the practical space requirement of some counters and experiences is much smaller than the theoretical requirement.

3.4 COCS Policy for Non-convex HFL

In this section, we will discuss the solutions for the client selection problems for non-convex HFL (e.g., neural network), where the convergence speed is quadratically related to the number of participated clients in each edge aggregation round [93, 159]. Similar to the utility function of strongly convex HFL in Eq. (3.6), the utility function of non-convex HFL is defined as follows:

$$\mu_{\text{non}}(\mathbf{s}^t; \mathbf{X}^t) = \sqrt{\frac{1}{M} \sum\nolimits_{m \in \mathcal{M}} \sum\nolimits_{n \in \mathbf{s}_m^t} X_{n,m}^t}. \tag{3.16}$$

Therefore, the client selection problem of non-convex HFL in edge aggregation round t is formulated as follows:

$$\text{P3:} \quad \max_{\mathbf{s}^t} \quad \mu_{\text{non}}(\mathbf{s}^t; \mathbf{X}^t) \quad \text{s.t. (3.8b), (3.8c), (3.8d).} \tag{3.17}$$

The problem P3 is also a combinatorial optimization problem. While brute-force search can always find the optimal solution, the complexity can be high due to the non-linear property in Eq. (3.17). In order to address this problem, we aim to design an efficient polynomial runtime approximation algorithm to solve P3 in the next subsection. In addition, the performance guarantee of the approximation algorithm will be presented.

3.4.1 Approximated Oracle Solutions

To solve the problem P3, we firstly show that P3 is a monotone submodular maximization problem with M knapsack and a matroid constraints. Below gives the definition of the monotone submodular maximization [75]:

Definition 1. *(Monotone Submodular Maximization.) A set function* $F : 2^I \to \mathcal{R}$ *is monotone increasing if* $\forall A \subseteq B \subseteq I$, $F(A) \leq F(B)$. *In addition, the function* $F(\cdot)$ *is submodular if* $\forall A \subseteq B \subseteq I$ *and* $e \in I \setminus B$, $F(A \cup \{e\}) - F(A) \geq F(B \cup \{e\}) - F(B)$.

Theorem 6. *P3 is a monotone submodular maximization with* M *knapsack and a matroid constraints problem.*

Proof. See in online Appendix D, available in the online supplemental material [127]. □

To facilitate the solution for non-convex HFL in P3, approximation algorithms are efficiently obtained approximate solutions in polynomial runtime. Some existing studies are focusing on solving the submodular maximization with knapsack and matroid constraints [10, 24], and they proposed the approximation guarantee to the optimal solution. In this chapter, we use the Fast Lazy Greedy (FLGreedy) algorithm in [10] to achieve the approximated oracle solution with $\frac{1}{(1+\epsilon)(2+2M)}$-approximation guarantee, where ϵ is an error parameter in the FLGreedy algorithm. If ϵ is small, FLGreedy can achieve a high approximation guarantee but have high computational complexity.

For coherence, we do not introduce the detailed FLGreedy algorithm here. The FLGreedy algorithm acquires the client selection decision of the client sequentially, which starts with the all-*null* decisions. In each edge aggregation round, it selects a client to an ES that gives the largest incremental learning utility. Because each client can only be selected at most one ES and each iteration decides the client selection decision for one client, the algorithm terminates in at most M iterations. Based on the results for submodular maximization with a knapsack and matroid constraints in [10], the FLGreedy algorithm guarantees to yield a $\frac{1}{(1+\epsilon)(2+2M)}$-approximation for P3:

Lemma 4. *In an arbitrary edge aggregation round* t, *let* $\mathbf{s}^{*,t}$ *be the client selection decision solved by FLGreedy algorithm and* $\mathbf{s}^{opt,t}$ *be the optimal client selection decision for the problem P3, we will have* $\mu(\mathbf{s}^{*,t}; \mathbf{X}^t) \geq \frac{1}{(1+\epsilon)(2+2M)} \mu(\mathbf{s}^{opt,t}; \mathbf{X}^t)$.

Proof. The proof follows [10] and hence is omitted. □

We use FLGreedy to approximate the optimal client selection decision with oracle information on participating clients. Note that the actual performance of the FLGreedy algorithm is usually much better than the $\frac{1}{(1+\epsilon)(2+2M)}$ approximation ratio in practice.

3.4.2 Performance of COCS Policy for Non-Convex HFL

The regret in Eq. (3.9) is used when the optimal oracle solutions can be derivable. Because the FLGreedy algorithm can efficiently approximate the optimal oracle solution for P3 instead of obtaining the optimal oracle solution. As such, we leverage the definition of δ-regret, which is usually used in MAB based on approximation algorithms [27]. For a δ-approximation algorithm (i.e., the solution $\mathbf{s}^t$ solved by the approximation algorithm satisfies $\mu(\mathbf{s}^t;\mathbf{X}^t) \geq \frac{1}{\delta}\mu(\mathbf{s}^{\text{opt},t};\mathbf{X}^t)$), for problem P3, the δ-regret is defined as follows:

$$R^{\delta}(T) = \sum\nolimits_{t=1}^{T} \frac{1}{\delta}\mu(\mathbf{s}^{\text{opt},t};\mathbf{X}^t) - \sum\nolimits_{t=1}^{T} \mu(\mathbf{s}^t;\mathbf{X}^t). \tag{3.18}$$

Because the FLGreedy algorithm for the P3 has an approximation ratio of $\frac{1}{(1+\epsilon)(2+2M)}$, COCS policy obtains $\delta = \frac{1}{(1+\epsilon)(2+2M)}$. The definition of δ-regret essentially compares the utility of a policy with the lower bound of the approximated oracle solution.

Next, we aim to present the input parameters $K(t)$ and h_T and propose a regret upper bound for the COCS policy. The regret analysis is also proved based on the Hölder condition in Assumption 5. Given the input parameters $K(t)$ and h_T in Theorem 7, we achieve a sublinear regret upper bound of COCS policy for P3 as follows:

Theorem 7. *(δ-Regret Upper Bound.) If $K(t) = t^{\frac{2\alpha+2}{3\alpha+2}}\log(t)$, $h_T = \lceil T^{\frac{1}{3\alpha+2}} \rceil$, Hölder condition holds true and a δ-approximation is applied for optimization, then the dominate order of δ-regret $\mathbb{E}[R^{\delta}(T)]$ is $O\left(\frac{4N^2MB}{\delta c^{min}} T^{\frac{2\alpha+2}{3\alpha+2}}\log(T)\right)$.*

Proof. See in online Appendix E, available in the online supplemental material [127]. □

The regret upper bound given in Theorem 7 implies that COCS policy performs well if the subproblem in each edge aggregation round can only be derived approximately. A sublinear δ-regret can be achieved by the performance guarantee of δ-approximation algorithms.

Table 3.1: HFL network parameters.

Parameter	Value
Number of clients, N	80
Number of ESs, M	3
Size of local model updates, s	0.18, 18.7, 31.3 (Mbits)
Computation workload, q^n	2.41, 28.3, 11.6 (Mbytes)
Transmission power, P_n^t	23dBm
Deadline, τ^{dead}	3, 20, 30 (sec)
Pricing function, $b_n(f_n)$	$\mathcal{U} \sim [0.5, 2]$ per Mbytes
Budget on each ES B	3.5, 40, 18
α in Hölder condition	1
h_T in COCS	5
Local training epochs, E	2, 5, 5
Global aggregation, T_{ES}	5
Learning rate, η	0.001, 0.1, 0.01

3.5 Simulations

3.5.1 Setup

1) Datasets and training models: We set up the simulation with PyTorch and the computation is conducted by a high-performance workstation with 2 NVIDIA RTX 2080 GPUs. We have prepared two datasets for evaluating our proposed COCS algorithm. Specifically, MNIST dataset [73] under a logistic regression, which is widely used for strongly convex FL studies [80, 183]. For the non-convex HFL, we use the CIFAR-10 dataset [69] by adopting a CNN with two 5×5 convolution layers (each with 64 channels), followed by 2×2 max polling, two fully-connected layers with 384 and 192 units, and finally a softmax output layer. The NLP dataset is built from the complete Works of William Shakespeare Shakespeare dataset [107]. We use a two-layer LSTM classifier with 100 hidden units and an 8D embedding layer. To this end, there is a densely-connected layer. For each simulation, we distribute the dataset among $N = 80$ clients in a general non-iid fashion such that each client only contains samples of only two labels.

2) Contexts: For the context generation, in each edge aggregation round, we assume the allocated bandwidth of all clients is sampling from a uniform distribution between $\mathcal{U} \sim$[0.3,

1] for MNIST dataset and $\mathcal{U}$ ∼[2, 4] for CIFAR-10 dataset since the data and model sizes are different for each dataset. Likewise, the available computation capacity of all clients is also sampling from $\mathcal{U}$ ∼[2, 4] for MNIST dataset, $\mathcal{U}$ ∼ [8, 15] for CIFAR-10 dataset, and $\mathcal{U}$ ∼[1, 2] for Shakespeare dataset. The distance $d_{n,m}^t$ between client and ES is from $\mathcal{U}$ ∼[0, 2]km. For SAFA, the distance d_n^t between clients and CS is from $\mathcal{U}$ ∼[0, 10]km.

3) Parameters of HFL networks: Our simulated HFL network includes 3 ESs and 50 clients, where the radius of each ES is 2km. Within the coverage area, there are several clients randomly distributed and communicated by the corresponding ES through a wireless channel in each edge aggregation round. In the edge aggregation round t, the downlink and uplink channel gain are decomposed of both small-scale fading and large-scale fading, where the small-scale fading is set as Rayleigh distribution with uniform variance and the large-scale fading are calculated by the path-loss with random shadowing $g_{\mathrm{DT},n}^t = g_{\mathrm{UT},n}^t = 37.6 \log(d_{n,m}^t) + 128.1$, where d represent the distance of client-ES pair (n, m). In order to clearly show the performance difference of each benchmark, we set the parameter of our simulated HFL network for the two datasets shown in Table 3.1.

3.5.2 Comparison Benchmarks

We compare the COCS policy to the following benchmarks: 1) Oracle: the Oracle algorithm knows precisely whether one client can be received by the corresponding ES before the deadline τ^{dead} with any observed context. In each aggregation round, it makes a client selection decision to maximize the utility in Eq. (3.8a): brute-force for the strongly convex HFL and GreedyLS for non-convex HFL. 2) Combinatorial UCB (CUCB): CUCB is designed based on a classical MAB policy UCB [7]. It develops combinations of client selection decisions on all ESs to enumerate NO's decision $\mathbf{s}$. CUCB runs UCB with feasible NO selection decisions $\mathbf{s}$ and learns the expected utility for each $\mathbf{s}^t$. Since CUCB does not fit the time-varying arm set, we set the static computation and transmission resource for client-ES pairs. 3) LinUCB: LinUCB [76] is a contextual variant of running CUCB. LinUCB also aims

to learn the expected utility for client selection decision $\mathbf{s}$, which assumes that the utility of an arm is a linear function of client-ES pairs' contexts. 4) Random: The Random algorithm selects a client to an accessible ES randomly in each edge aggregation round under these two constraints. 5) SAFA: SAFA [168] is an asynchronous FL algorithm. If some clients cannot be received before the deadline, they will join the next aggregation stage.

3.5.3 Performance Evaluation of Strongly Convex HFL

1) Comparison of cumulative utilities: Figures 2 show the cumulative utilities and regret obtained by the COCS policy and the other 4 benchmarks during 1,000 edge aggregation rounds under logistic regression on the MNIST dataset. For the cumulative utilities in Figure 3.3, it is observed that Oracle policy achieves the highest cumulative utilities and provides an upper bound to the other benchmarks as expected. Among the others, COCS policy significantly outperforms the other benchmarks and has a closed cumulative utility performance to Oracle policy. The profit of the context of client-ES pairs can be shown by comparing the performance of context-aware policies (LinUCB) and context-unaware policies (CUCB and Random). More specifically, the results show that the cumulative utilities of CUCB are similar to the Random policy. The disadvantage of CUCB comes from the following two reasons: (1) an arm of CUCB is a combination of the selection decisions of all client-ES pairs, and hence CUCB obtains a large number of arms. This means that CUCB is difficult to enter the exploitation phase. (2) CUCB fails to capture the connection between context and clients. The cumulative utilities of LinUCB are based on the context, for which a CUCB arm is not effective to produce a good result due to the large arm set. The reason that the cumulative utilities of SAFA are low (only higher than random) is that the average distance between clients and CS is much larger than HFL. Therefore, due to the low value of τ^{dead}, most clients cannot be received within one aggregation round. In Figure 3.3, we notably depict the regret generated by the 4 benchmarks. It is easy to observe that our proposed COCS policy incurs a sublinear regret.

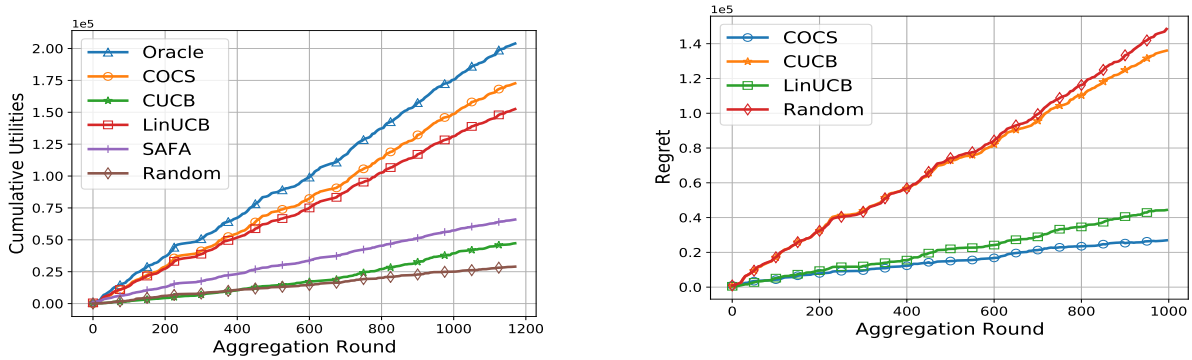


Figure 3.3: Cumulative utilities and regret on MNIST.

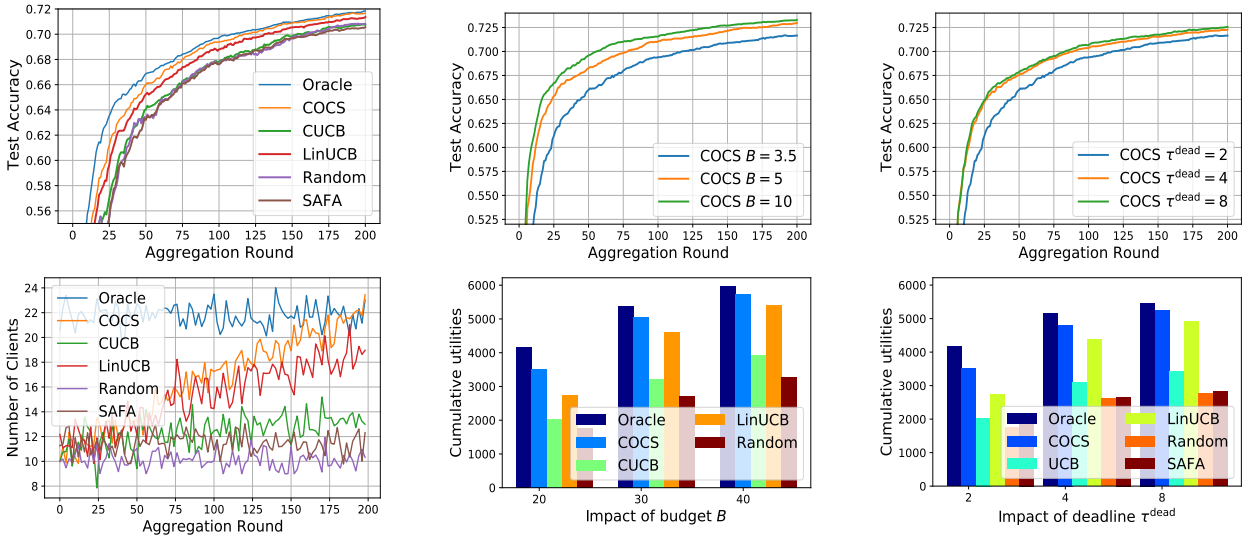


Figure 3.4: Training performance on MNIST.

Table 3.2: Final accuracy and edge aggregation rounds.

Policy	MNIST (70%)		CIFAR-10 (60%)		Shakespeare (45%)	
	Final Accuracy	Round	Final Accuracy	Round	Final Accuracy	Round
Oracle	71.90 (74.85)	111 (70)	68.41 (73.81)	92 (58)	58.83 (61.24)	223 (180)
COCS	71.84 (74.57)	121 (74)	67.93 (73.12)	101 (63)	57.66 (60.39)	246 (192)
UCB	71.15 (73.53)	147 (86)	63.76 (68.16)	175 (120)	54.25 (57.19)	289 (219)
LinUCB	71.67 (74.26)	134 (79)	65.19 (72.30)	133 (82)	55.52 (59.87)	251 (201)
Random	70.81 (72.31)	161 (103)	62.25 (67.79)	207 (146)	47.30 (52.41)	339 (297)
SAFA	71.07 (73.40)	152 (93)	63.31 (66.93)	184 (129)	53.63 (56.18)	296 (230)

2) Training performance and client selection results: We use two metrics to evaluate the training performance based on client selection policies: edge aggregation rounds to achieve targeted testing accuracy and final testing accuracy. In Figure 3.4, we present the training performance under logistic regression on the MNIST dataset. Oracle policy, as expected, re-

Table 3.3: Final accuracy and edge aggregation rounds without Z constraint.

Policy	MNIST (70%)		CIFAR-10 (60%)		Shakespeare (50%)	
	Final Accuracy	Round	Final Accuracy	Round	Final Accuracy	Round
Oracle	71.90 (74.85)	111 (70)	68.41 (73.81)	92 (58)	58.76 (61.15)	227 (181)
COCS	71.20 (74.16)	126 (77)	66.10 (71.35)	109 (71)	57.13 (60.01)	251 (198)
UCB	69.98 (72.01)	181 (109)	61.54 (64.27)	196 (133)	53.11 (56.02)	311 (236)
LinUCB	70.97 (73.91)	142 (86)	64.43 (71.04)	146 (93)	55.03 (57.93)	259 (210)
Random	68.16 (70.03)	208 (149)	60.48 (63.20)	276 (208)	43.49 (47.93)	386 (348)

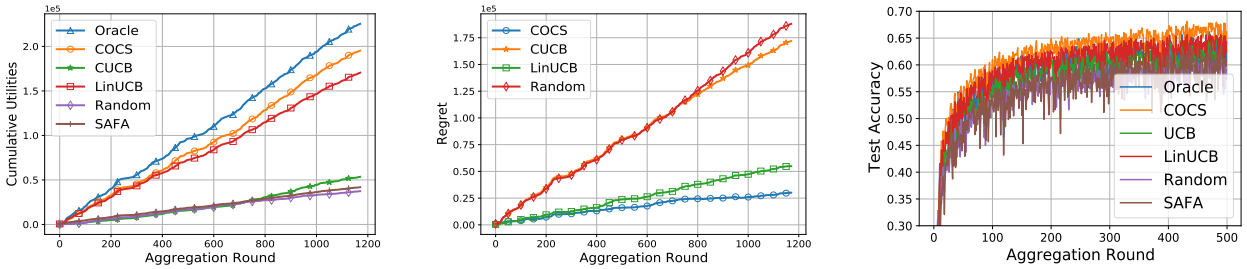


Figure 3.5: Training performance on CIFAR-10.

sults in the fastest convergence speed and highest accuracy among all benchmarks. Although COCS performs slower than Oracle policy during the first several rounds due to exploration, it achieves similar testing accuracy to Oracle in the 200th round. In particular, it is easy to observe that COCS outperforms others. Due to the insufficient selection of clients all the rounds, Random selection is considerably inferior to all other benchmarks. For clarifying the training performance, we present an auxiliary Table 3.2 to emphasize the results, in which the targeted accuracy on the MNIST dataset is set 70%. As is shown, our proposed COCS policy only uses 121 rounds to achieve 70% test accuracy, which is 36, 13, 40, and 38 rounds faster than CUCB, LinUCB, Random, and SAFA.

In Figure 3.4, we show that the temporal number of clients is selected in each edge aggregation round. The upper bound and lower bound of clients are Oracle and Random policies. Although COCS, LinUCB, and CUCB all have increasing levels due to obtaining historical experiences, it is easy to observe that the increase of CUCB is very slow, and COCS outperforms the other two benchmarks. The reason why the number of successful participated clients is few in the first several rounds via the MAB-based policies is that most of the selected clients cannot be received by ESs before the deadline τ^{dead}.

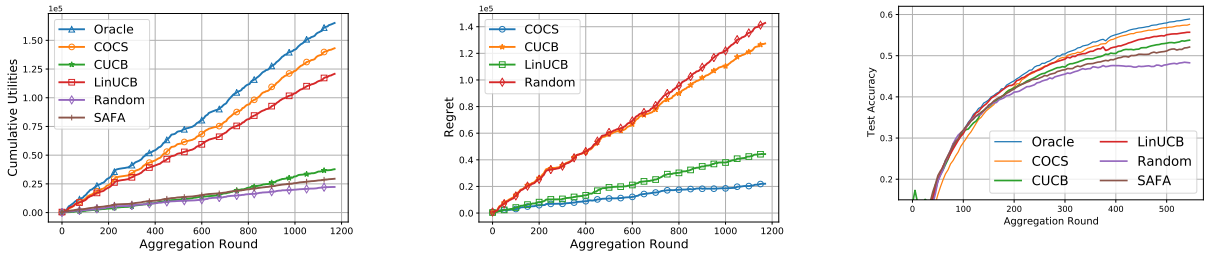


Figure 3.6: Training performance on Shakespeare.

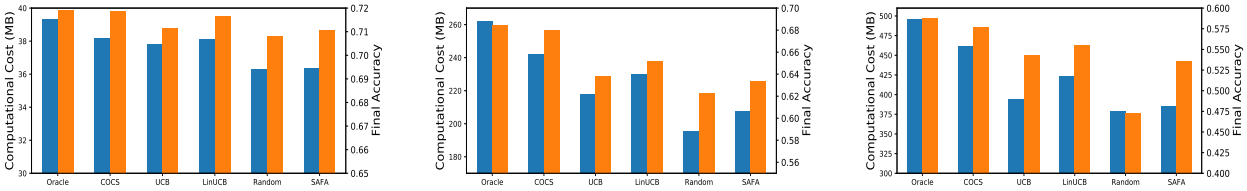


Figure 3.7: Computational cost on the three datasets.

3) Impact of budget B: Figure 3.4 shows that the training performance of COCS under different budgets B ($B = 3.5$, 5 and 10). It is easy to observe that COCS has better performance when NO increases the budget B. This is simply to explain because increasing the budget can select more clients in each round to increase the utility of HFL. In particular, when $B = 5$ NO only uses 77 rounds to achieve 70% test accuracy, which is much faster than $B = 3.5$. However, the training performance does not have a significant improvement from $B = 5$ to 10. Figure 3.4 presents the cumulative utilities of these five benchmarks after 200 edge aggregation rounds. Clearly, for all the benchmarks, the NO can achieve higher cumulative utilities with an increasing budget. As can be observed, when the budget increases from 3.5 to 5, the benefit of client selection gradually increases. However, although the payment budget is set large, the redundant selection does not have significant improvement. The reason may be because some client-ES pairs performing poor transmission status cannot be successfully received by ESs before the deadline τ^{dead}, even if NO increases the budget of computation resources.

4) Impact of deadline τ^{dead}: Figure 3.4 depicts the training performance and Figure 3.4 depicts the cumulative utilities under different deadlines $\tau^{\text{dead}} = 2$, 4 and 8. We can see

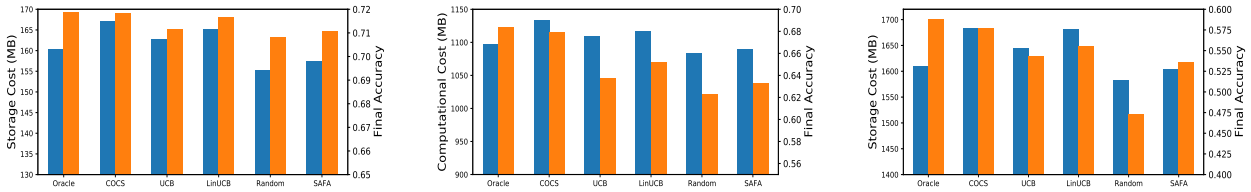


Figure 3.8: Storage cost on the three datasets.

that when NO increases the value of the deadline, the number of clients increases gradually, which performs similarly to increasing budget B. However, if NO sets the deadline too large (i.e., $\tau^{\text{dead}} = 8$), training performance and cumulative utilities perform similarly to $\tau^{\text{dead}} = 4$. More specifically, cumulative utilities of COCS only increase from 5,125 to 5,378. It is easy to observe that the impact of increasing τ^{dead} is less than increasing budget B. We consider that less budget can control the number of selected clients, which is more dominant than the impact of deadline τ^{dead} for the training performance both on convergence and accuracy.

3.5.4 Performance Evaluation of Non-convex HFL

We show the performance of non-convex HFL under the CNN model on the CIFAR-10 dataset and LSTM model on the Shakespeare dataset, where the utility is quadratically related to the number of participated clients. We set the error parameter of FLGreedy algorithm $\epsilon = 0.3$. Figure 3.5 depicts the cumulative utilities and Figure 3.5 depicts regret. Similar to the performance of strongly convex HFL, Oracle policy performs the best cumulative utilities as expected, and COCS outperforms the other 3 benchmarks (e.g., 1.7× higher than CUCB policy). In particular, the difference of cumulative utilities between Oracle and COCS is smaller than the result, since this is an approximated Oracle solution in this case. It is also observed that COCS achieves a sublinear regret in Figure 3.5. Figure 3.5 shows the test accuracy of different client selection benchmarks on the CIFAR-10 dataset. Since the training model and data size of CIFAR-10 is complicated enough, training performance of different benchmarks are clear to see. Oracle policy has the best performance among all benchmarks, which achieves 68.41% test accuracy. In Table 3.2, COCS policy can achieve

67.93% test accuracy, which is 4.17%, 1.74%, 5.68%, and 4.52% higher than LinUCB, CUCB, Random, and SAFA policies.

Figure 3.6 shows the cumulative utilities, regret, and training performance on the Shakespeare dataset under LSTM. The results are similar on other datasets, where COCS consistently has the best performance except Oracle policy, and Random policy performs worst. From the convergence performance perspective, COCS also outperforms the other three benchmarks. We can conclude that COCS policy can improve the training performance for HFL significantly both in strongly convex and non-convex settings.

3.5.5 Other Simulation Results

To present the efficiency of computation and storage, we show the trade-off between computational cost and final accuracy in Figure 3.7, and the trade-off between storage cost and final accuracy in Figure 3.8. It is worth noting that both two costs include training cost, edge aggregation cost, and client selection cost. It means that if NO selects more clients, the cost should increase. For example, on the Shakespeare dataset in Figure 3.7, COCS selects 17.29 more clients than the Random policy, but only increases 57.36MB, which indicates that the client selection policy does not waste much computational cost. In addition, since the large computational cost for solving the results of Oracle policy by the greedy algorithm, the computational cost is high enough. In summary, we consider that making client selection does not have an obvious impact on training costs.

3.6 Related Work

Clients selection can efficiently deal with the straggler problems and significantly improve the performance of FL in terms of convergence speed and training latency. For example, [158] designs a deep reinforcement learning algorithm (the local model updates and the global model are considered as states) to select clients. [111] uses gradient information to select clients. If the inner product between the client's local and global gradient is negative,

it will be excluded. In [141], they develop a system model to estimate the total number of aggregation rounds and design a greedy algorithm to jointly optimize client selection and bandwidth allocation throughout the training process, thereby minimizing the training latency. [176] designs a dynamic client sampling method. In the early aggregation rounds, fewer clients are selected, and more clients are in later rounds. This method has been proven to improve training loss and accuracy as well as decrease overall energy consumption.

HFL has been considered to be a more practical FL framework for the current MEC system since the hierarchical architecture makes FL communication more efficient and significantly reduces the impact of straggler [92]. Later, some studies improve the performance of HFL from different perspectives or use it in some other applications. For example, [93, 159] propose a detailed convergence analysis of HFL, showing that the convergence speed of HFL achieves a linear speedup of conventional FL. Recently, FL has attracted more interest, especially with the rapid development of ML applications on IoT devices. [81] designs a hierarchical blockchain framework for knowledge sharing on smart vehicles, which learns the environmental data through ML methods and shares the learning knowledge with others. [167] uses HFL toadapt to personalized modeling tasks and protect private information.

The MAB problem has been extensively studied to address the key trade-off between exploration and exploitation making under uncertain environment [130], and it has been used in FL for designing the client scheduling or selection [170, 55, 165]. For example, [29] considers a MAB problem for edge service provisioning with budget constrained and [28] studies the optimal sniffer channel assignment for small cell cognitive radio networks. It has also been widely used in FL for designing the client scheduling or selection [170, 55, 165]. [170] designs a client scheduling problem and provides a MAB-based framework for FL training without knowing the wireless channel state information and the dynamic usage of local computing resources. To minimize the latency, [55] models fair-guaranteed client selection as a Lyapunov optimization problem and presents a policy based on CC-MAB to estimate the model transmission time. A multi-agent MAB algorithm is developed to minimize the

FL training latency over wireless channels, constrained by training performance as well as each client's differential privacy requirement in [165]. In this chapter, the COCS policy is proposed to select clients for HFL. In FL, CS connects all clients and the available set of selecting clients does not change in each aggregation round. However, in HFL, due to the dynamic connection conditions of the client-ES pair and the limited available computing capacities of clients in each edge aggregation round, we cannot assume that each ES can make a selection decision for the same clients set, which indicates that the COCS policy must face with two constraints for deciding which clients can be selected and how to rent the computational resources. This two constraint can be divided into two different categories: knapsack and matroid constraints rather than single constraint MAB problem [29, 28], which brings more challenges than these studies.

3.7 Summary

In this chapter, we investigated the client selection problem to improve the training performance for HFL. An online decision-making policy, called Context-aware Online Client Selection (COCS), was designed for NO to make proper client selection decisions for each client-ES pair. COCS was developed based on the CC-MAB framework, where NO observes the context (i.e., downloading channel state and local computation resources) of client-ES pairs and learns the participated probabilities to select clients and guide rental computation resources. Our proposed COCS policy departs from conventional optimization-based algorithms, which can work in HFL networks with uncertain information. More specifically, COCS addresses many practical challenges for HFL networks, and it is easy to implement and achieves a provably asymptotically optimal performance guarantee. Although our COCS policy has presented a superior performance in extensive HFL experiments, there are still several future research questions. For example, considering the dynamic partition of context space may improve the selection results, since it can generate more appropriate hypercubes. A theoretical improvement of convergence speed from COCS should also be considered.

Chapter 4: LoMar: A Local Defense Against Poisoning Attack on FL

4.1 Abstract

Federated learning (FL) provides a highly efficient decentralized machine learning framework, where the training data remains distributed at remote clients in a network. Though FL enables a privacy-preserving mobile edge computing framework using IoT devices, recent studies have shown that this approach is susceptible to poisoning attacks from the side of remote clients. To address the poisoning attacks on FL, we provide a two-phase defense algorithm called Local Malicious Factor (LoMar). In phase I, LoMar scores model updates from each remote client by measuring the relative distribution over their neighbors using a kernel density estimation method. In phase II, an optimal threshold is approximated to distinguish malicious and clean updates from a statistical perspective. Comprehensive experiments on four real-world datasets have been conducted, and the experimental results show that our defense strategy can effectively protect the FL system. Specifically, the defense performance on the Amazon dataset under a label-flipping attack indicates that, compared with FG+Krum, LoMar increases the target label testing accuracy from 96.0% to 98.8%, and the overall averaged testing accuracy from 90.1% to 97.0%[3].

4.2 Background and Problem Formulation

4.2.1 Federated Learning

FL systems are developed to train a joint model $\mathbf{w}$ to address a distributed optimization problem (e.g., image classification) in a decentralized network. Typically, an FL system

[3]This chapter was accepted in IEEE Transactions on Dependable and Secure Computing, November 2021. Permission is included in Appendix D.

is composed of N remote clients and one aggregator. Considering that the whole training data in the FL system is $\mathcal{D}$ and each remote client has its private training dataset (e.g., $\mathcal{D}_i$ for the i-th client), we have $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_i, \dots, \mathcal{D}_N\}$, where the size of i-th private dataset is denoted as $l_i = |\mathcal{D}_i|$ and the total number of training samples is $l = |\mathcal{D}| = \sum_i^N l_i$. Different from the conventional learning methods, the training data in FL is only stored and processed on remote clients. Meanwhile, the aggregator maintains a joint model and repeatedly updates it with the received local learning updates from remote clients according to a certain aggregation rule. At the beginning of an FL process, the aggregator initializes a joint model $\mathbf{w}^0$ and distributes it to all remote clients for local training. Specifically, at the t-th iteration, the FL system repeats the following three steps to obtain the joint model $\mathbf{w}^t$ from the current $\mathbf{w}^{t-1}$.

1) The aggregator delivers the joint model $\mathbf{w}^{t-1}$ to all remote clients.

2) Each remote client performs its local learning process with its local training data and the received joint model $\mathbf{w}^{t-1}$. During the local learning process, an updated local model $\mathbf{w}_i^t$ is produced by the stochastic gradient descent as $\mathbf{w}_i^t = \mathbf{w}^{t-1} - \eta_i \nabla L_i(\mathbf{w}^{t-1}, \mathcal{D}_i)$, where η_i is the learning rate of local model training and $\nabla L_i(\mathbf{w}^{t-1}, \mathbf{D}_{l_i})$ denotes the gradient of local optimization loss. Once the local training is finished (e.g., with several local epochs), each client sends back the local training update $\mathbf{u}_i^t = \mathbf{w}_i^t - \mathbf{w}^{t-1}$.

3) The aggregator aggregates all local model updates to obtain the updated joint model $\mathbf{w}^t$ using a specific aggregation rule, e.g., the typical weighted averaging rule (FedAvg), given by $\mathbf{w}^t = \mathbf{w}^{t-1} + \sum_{i=1}^M \alpha_i \mathbf{u}_i^t$,where $\alpha_i = \frac{l_i}{l}$. Usually, the aggregation rule guarantees an optimized solution for the distributed optimization problem in the FL system.

4.2.2 Attacker

In general, there are two types of poisoning attacks in the machine learning field: targeted poisoning attacks and non-targeted poisoning attacks. Though the ultimate goal for both types can be considered as decreasing the performance of the FL learned joint model, the

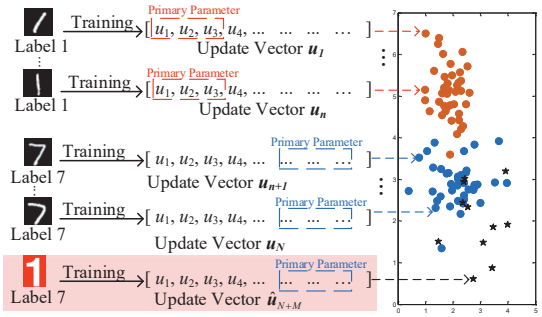


Figure 4.1: Model updates under attack on MNIST.

implementation details can vary. For example, the attacker in targeted poisoning attacks typically aims to take control of a specific ratio of the learning objective (e.g., the targeted labels in the image classification). Moreover, some of these attacks develop their attack mechanisms based on a trade-off strategy to make the ML model converge on the other labels while successfully controlling the target label. In this situation, the attacks are more difficult to be detected. And for the non-targeted poisoning attacks, they usually manipulate the ML model such that it would have a high error rate for testing data samples and be unusable for further learning tasks, which do not target a specific classification label.

As FL gets more popular in recent years, the security problems in the FL systems gain much interest from the ML community. Especially, compared to centralized learning models, implementing an attack on the FL system is much easier because of its loose structure and plenty of space between the remote clients and the aggregator. Therefore, multiple poisoning attacks have been developed to break the FL systems, most of which describe their attacking objective as an optimization problem and manipulate the joint model by sending poisoned remote updates. For simplicity, we denote the joint model from a clean FL system as $\mathbf{w}$ while the poisoned is represented as $\hat{\mathbf{w}}$ in the rest of this chapter. Note that for better presentation, the attacker introduced in the following chapter is considered a targeted poisoning attack, which intends to manipulate one specific label.

Usually, the attackers can produce the poisoned local training updates by injecting new malicious clients into the system or manipulating original clean clients. In this chapter, we

consider the former scenario to present the notations. Note that the capacity of the attacker in this chapter is regulated as follows: i) the attacker injects M malicious clients into the FL system, where the budget of M is $M \leq 0.4N$; ii) by default, the number of target label is considered as one. Hence, we denote the malicious training dataset, models, and updates are denoted by $\mathbf{D}_m$, $\mathbf{w}_m$, and $\mathbf{u}_m$, respectively.

4.2.3 Defender

To address the poisoning attacks, we consider the ultimate goal of the defender to successfully remove the malicious impact from the FL system. In general, there are two existing types of mechanisms to develop a defender: i) one is to detect the malicious based on the analysis of remote updates to the joint model (i.e., Auror[139]); ii) the other is to develop a new aggregating function to achieve the FL learning objective with the poisoned remote updates, which is also known as Byzantine tolerance (e.g., Krum [185]). However, existing defense mechanisms are proven to have weak performance against poisoning attacks on distributed machine learning models, especially on FL. For example, as shown in [18], the baseline label-flipping attack can bypass most of the existing defense approaches by adding a simple stealth metric.

After reviewing existing poisoning attacks and defense mechanisms on FL, we summarize the reason why current defenses fail. According to the principle of parsimony, a common sense ML attack is that a successful attacker leverages an efficient yet effective impact on the learning model. This indicates that compared to the clean benign updates, the poisoned remote updates share a unique difference in their attack objectives. Intuitively, the poisoning attacks on an FL system can be easily addressed if the defender finds a feasible measurement to distinguish the clean and poisoned updates. However, due to the complexity (the huge number of layers and high dimension of the input information) of the remote updates, developing a precise criterion from a global view and considering the malicious as a global anomaly can lead to the failure of developing a successful defense.

For instance, we introduce an observation of remote updates under label flipping 1 - 7 attack on the MNIST dataset with a logistic regression classifier in Figure 4.1, where the defender needs to identify the malicious updates on label 1. However, the defense would fail as the malicious updates may locate close to clean ones by only manipulating a small size of parameters, which makes them hard to be detected.

To address this challenge, it is necessary to develop a new FL defense algorithm with a feasible measurement of the remote updates for better maliciousness detection. As such, we propose a new defender, called the Local malicious factor (LoMar). Different from the existing defense methods, LoMar provides its malicious detection strategy based on the parameter features of the remote updates, which is considered to be a local criterion, instead of from a global view. Specifically, LoMar uses a scored factor $F(i)$ to denote the degree of maliciousness for each remote client and develops a new aggregation rule on the aggregator side, which comes with a binary controller $\delta(i)$ to protect the joint model from poisoned updates. Formally, we describe the new aggregation rule of the FL system under the protection of LoMar as follows:

$$\tilde{\mathbf{w}}^t = \tilde{\mathbf{w}}^{t-1} + \sum_{i=1}^{N+M} \delta(i)\alpha_i \mathbf{u}_i^t. \tag{4.1}$$

where $\tilde{\mathbf{w}}^t$ denotes the joint model in a potentially poisoned FL system with LoMar protection.

4.3 Design of LoMar Defense

4.3.1 Overview

Our motivations for developing the two-phase LoMar defense on the FL system come from two main challenges: i) finding the local density distribution of the remote update from the parameter perspective can be very difficult and expensive; ii) it is also extremely hard to find an appropriate approach for the malicious detection of each remote update, i.e., the clean updates will be considered as malicious if the defense mechanism is too strict and the poisoning updates cannot be fully detected if it's too loose. Moreover, it is unrealistic for the defender to obtain a clean joint model $\mathbf{w}$ for further analysis because the FL system

can not be trusted from the start of the FL training process as it might have been already poisoned. In order to address these challenges, intuitively, although the true distribution of each remote update is not available in the FL, we can obtain a local density estimation for the i-th remote update as $\tilde{q}(\mathbf{u}_i)$, based on which all possible poison attacks can be then detected using the local outlier detection method.

In phase I, for the i-th remote update at the t-th iteration, LoMar finds its k-nearest neighbors to be its neighborhood reference set $\mathcal{U}_i = \{\mathbf{u}_{i,1}, \dots, \mathbf{u}_{i,k}\}$, where $k \ll N + M$. Achieving this in an FL system can be extremely difficult, as the probability distribution of remote clients is not known to the aggregator. As such, we use a state-of-art non-parametric estimation method, which is the kernel density estimation (KDE) [148], to estimate the local density distribution function for $\mathbf{u}_i$ according to each output label in the FL joint model. Specifically, in this chapter we assume an extremely difficult scenario that the defender has no idea about the number of malicious in the FL system. Thus, we pursue an alternative solution that defines the value of $k \leq 0.4N$ to be a loose bound, corresponding to the maximum number of potentially malicious. Particularly, in the recent studies of data poisoning attacks against FL, the number of data poison is considered an important constraint. For example, [56] sets an upper bound of the malicious budget at only 10%. And work in [43] shows that if the size of malicious is up to 50%, the learning speed of the FL network can be very slow that making the attack easy to be found. Then, we define the malicious factor $F(i)$ to describe the numerical malicious degree for the i-th client. We introduce the process of the KDE estimation in LoMar in Figure. 4.2: i) the remote updates are divided by the dimension of the parameter features in the pre-processing process; ii) the parameters are clustered based on the output labels; iii) the kernel density estimation of each output label is performed and being a product to one numerical output.

In phase II, we provide a boolean vector to determine the malicious status of an uncertain remote update as 1 (clean) or 0 (malicious). In this procedure, a threshold ϵ is developed from an inequality approach to manipulate the numerical factor $F(i)$ in phase I into the boolean

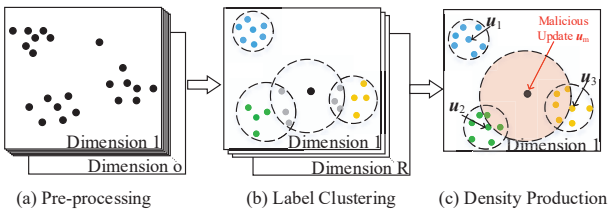


Figure 4.2: KDE estimation structure in LoMar.

output $\delta(i)$. Theoretical analysis is also performed regarding the selection of threshold in terms of false alarm rate.

4.3.2 Phase I: Malicious Client Factor

In a FL system, the remote updates set at the t-th iteration is $\mathcal{U} = \{\mathbf{u}_1^t, \mathbf{u}_2^t, \dots, \mathbf{u}_{N+M}^t\}$. Note that in FL, the i-th remote update $\mathbf{u}_i^t$ is always related to the iteration number, and thus in the rest of this section, we simplify the $\mathbf{u}_i^t$ to $\mathbf{u}_i$, where the superscript of $\mathbf{u}_i$ would be denoted to present the proposed LoMar algorithm. The process of finding the malicious client factor $F(i)$ comes from two steps: finding the k-nearest neighborhoods and developing $F(i)$ with the neighborhood from the local KDE. For the update $\mathbf{u}_i$, we develop a neighborhood reference set $\mathcal{U}_i = \{\mathbf{u}_{i,1}, \dots, \mathbf{u}_{i,j}, \dots, \mathbf{u}_{i,k}\}$, which collects the k-nearest neighborhoods of each update instead of the whole remote updates. To evaluate their relative positions to the kernel center $\mathbf{u}_i$, we calculate the averaged l_2 norm distance between $\mathbf{u}_i$ to neighbors in $\mathcal{U}_i$, where the averaged distance is defined as $\bar{d}_i = \frac{1}{k}\sum_{j=1}^{k} \mathsf{diff}(\mathbf{u}_i, \mathbf{u}_{i,j})$. To further explain this definition, the function $\mathsf{diff}(\cdot,\cdot)$ leverages a squared Euclidean distance $||\cdot||^2$ between $\mathbf{u}_i$ and its neighbor. Additionally, we further investigate the difference of all updates in $\mathcal{U}_i$ to represent the mean value of $\bar{d}_i$ for each $\mathbf{u}_i \in \mathcal{U}_i$ as $\bar{D}_i = \frac{1}{k(k-1)}\sum_{i=1}^{k}\sum_{j=1}^{k} \mathsf{diff}(\mathbf{u}_i, \mathbf{u}_{i,j}), \quad i \neq j$.

Intuitively, in low density areas, $\bar{d}_i$ is large and the ratio $\frac{\bar{d}_i}{\bar{D}_i}$ will spread out. In high-density areas, the result will be reversed. Previous studies [96, 22] have shown that the k-nearest neighborhood mapping is adaptive to estimate the local sample density from the distance perspective.

As we mentioned in Section 4.2, the objective of the FL system is to train a joint model $\mathbf{w}$ for a distributed learning problem. Note that a typical machine learning problem usually comes with multiple output labels, e.g., MNIST which aims to recognize digits 0-9. Therefore, it is obvious that for each $\mathbf{u}_i$, the updated parameters from remote clients make different contribution to the output, which could be denoted as $\mathbf{y} = [y_1, ..., y_r, ... y_R]$, where $r \in [1, R]$ denotes the r-th output label. Considering the relationship between $\mathcal{U}_i$ and $\mathbf{y}$ to be a probabilistic graph model, then we could use the expected probability distribution of each output label, e.g., $\mathbb{P}(y_r) = \mathbb{P}(y_r|\mathcal{U}_i)$ for the r-th output, to approach the joint distribution $\mathbb{P}(y) = \prod_{r=1}^{R} \mathbb{P}(y_r)$. Therefore, through our defender still has no access to the true statistic distribution of $\mathbf{u}_i$, we can approach an estimated probability distribution based on the analysis of each output label that satisfies $\sum_{r=1}^{R} \mathbb{P}(\mathbf{u}_i|y_r) = 1$. In this chapter, we use a non-parametric kernel density estimation function to approach the estimated local distribution for the i-th remote update $\mathbf{u}_i$ with its k-nearest neighborhood $\mathcal{U}_i$. Specifically, the estimated distribution of i-th client on r-th output label is denoted as $\tilde{q}(\mathbf{u}_i^r)$. Formally, the process of obtaining the $\tilde{q}(\mathbf{u}_i^r)$ can be formulated by:

$$\tilde{q}(\mathbf{u}_i^r) = \frac{1}{k} \sum_{j=1}^{k} K\left(\frac{\mathbf{u}_i^r - \mathbf{u}_{i,j}^r}{h^r}\right), \tag{4.2}$$

where $K(\cdot)$ is the kernel function and $\mathbf{u}_{i,j}^r$ denotes the r-th output from the j-th neighbor in $\mathcal{U}_i$. And h is the bandwidth of $K(\cdot)$, which is a user-defined constant value that is correlated to the learning classifier. Note that theoretically, the value of h could be different for each output label. For better presentation, we consider it to be constant in this chapter. Particularly, we consider the $K(\cdot)$ is a Gaussian function as $K\left(\frac{\mathbf{u}_i^r - \mathbf{u}_{i,j}^r}{h}\right) = \frac{1}{(\sqrt{2\pi})h} \exp\left(-\frac{|\mathbf{u}_i^r - \mathbf{u}_{i,j}^r|}{2h}\right)$. Based on the estimated $\tilde{q}(\mathbf{u}_i^r)$, we define the maliciousness factor $F(i)^r$ for the i-th remote update for r-th output label in its neighborhoods as follows:

$$F(i)^r = \frac{\sum_{j=1}^{k} \tilde{q}(\mathbf{u}_j^r)}{k\tilde{q}(\mathbf{u}_i^r)}, \tag{4.3}$$

where $\mathbf{u}_j^r, j \in [1, k]$ is the local neighbors of the i-th remote update on the r-th label. Therefore, for each label, we could obtain a corresponding maliciousness degree factor $F(i)^r$ for

the i-th remote update. We then estimate the numerical maliciousness degree factor $F(i)$ of $\mathbf{u}_i$ with the obtained $\mathbf{u}_i^r$ according to each output label. Note that for two different outputs r and r', we consider the probability distribution of $\mathbf{u}_i^r$ and $\mathbf{u}_i^{r'}$ in the same neighborhoods $\mathcal{U}_i$ could satisfy the following relationship that $\mathbb{P}(\mathbf{u}_i^r, \mathbf{u}_i^{r'}|y_r, y_{r'}) = \mathbb{P}(\mathbf{u}_i^r|y_r) \cdot \mathbb{P}(\mathbf{u}_i^{r'}|y_{r'})$. Then the joint probability of $\tilde{\mathbb{P}}(\mathbf{u}_i)$ is: $\tilde{\mathbb{P}}(\mathbf{u}_i) = \prod_{r=1}^{R} \mathbb{P}(\mathbf{u}_i^r)$.

Definition 2. *From the relationship between each label $\mathbf{u}_i^r$ and the remote update $\mathbf{u}_i$, we give the definition $F(i)$ from the obtained $F(i)^r$ that*

$$F(i) = \frac{\sum_{j=1}^{k} \tilde{q}(\mathbf{u}_j)}{k\tilde{q}(\mathbf{u}_i)} = \frac{\sum_{j=1}^{k} \prod_{r=1}^{R} \tilde{q}(\mathbf{u}_j^r)}{k\tilde{q} \prod_{r=1}^{R} (\mathbf{u}_i^r)} = \prod_{r=1}^{R} F(i)^r. \tag{4.4}$$

Explanation. See Appendix B.1 in detail. □

Note that the obtained $F(i)$ can be considered as the same as $\frac{\bar{d}_i}{\bar{D}_i}$ from the statistical perspective: the negative exponential operator of kernel function indicates that the value of $\bar{d}$ becomes smaller when the difference between $\bar{d}$ and $\bar{D}$ grows larger. Specifically, we consider the higher value of $F(i)$ is, the relative position of $\mathbf{u}_i$ is closer to the kernel center of its local neighborhood. $F(i)$ can lead to a more accurate estimation when the value of h is customized for each dimension in each cluster and we set this parameter as a constant value in our chapter for simplicity, as finding the optimal hyperparameter is not the goal of this chapter. The process to obtain the maliciousness degree factor vector $F(i)$ for the remote client. For convenience, we assume that the output labels are already known at the beginning of the LoMar defense. In summary, we use LoMar to obtain the malicious factor in the following two steps: i) developing the k-nearest neighborhood map; ii) obtain $\tilde{q}(\mathbf{u}_i^r)$ and calculate the malicious factor vector $F(i)$.

4.3.3 Phase II: Finding Decision Threshold

After the phase I of LoMar, we obtain the numerical malicious factor vector $F(i)$ for each remote client. In order to achieve the optimal defense goal in Eq. (4.1), LoMar develops a binary controller with a feasible threshold ϵ to turn the numerical results $F(i)$ into boolean

status $\delta(i)$ to finally identify and remove malicious clients. Formally, if $F(i) \geq \epsilon$, we consider the remote client to be clean, otherwise malicious if $F(i) < \epsilon$.

However, there are several issues in the process of finding ϵ in the poisoned FL system: i) although the updates share similar features according to specific classifiers, the true distribution of $\mathbf{u}_i$ is still unavailable; ii) the threshold ϵ can hurt the learning objective of FL if it is too strict. To address these issues, we provide an asymptotic threshold of ϵ for our LoMar, which aims to determine most of the malicious remote updates with a high positive malicious detection rate to protect the clean updates in FL.

Firstly, we discuss the lower bound of ϵ_m for malicious updates in an FL system under poisoning attacks. In particular, the following theorem tells the asymptotic false alarm rate for a given ϵ_m:

Theorem 8. *Assuming that the lower bound of the probability distribution of malicious updates is ϵ_m, then the possibility of false detecting a clean update $\mathbf{u}_i$ as malicious would be*

$$P(F(i) > \epsilon_m) \leq \exp\left(-\frac{4\pi(\epsilon_m - 1)^2(k+1)^2\bar{h}}{k(2k+\epsilon_m+1)^2V^2}\right), \tag{4.5}$$

where k is the size of its k-nearest neighborhoods, V denotes the neighborhood kernel density distribution volume of $\mathbf{u}_i$ and $\bar{h}$ indicates the average bandwidth of $\mathbf{u}_i$ (which is considered as a constant in this chapter).

Proof. See Appendix B.2 in detail. □

Then, we consider an optimal condition that there is a trusted FL system, which consists of only clean remote clients. In this scenario, there exists a statistical boundary for the expectation of $F(i)$ for the furthest clean update $\mathbf{u}_i$ to the kernel center. The following theorem shows how to determine the boundary of clean remote updates.

Theorem 9. *Consider that the remote update $\mathbf{u}_i$ is generated from a continuous distribution in a FL system. Then, when $N \to \infty$, there exist a boundary σ that $F(i) = 1$ where the remote update $\mathbf{u}_i$ is considered to be clean (not abnormal) as $\sigma = \frac{\mathbb{E}(\bar{D}_i)}{\mathbb{E}(\bar{d}_i)} = 1$.*

Proof. Considering the situation that the updates are uniformly distributed around $\mathbf{u}_i$, then the discrete space $\mathcal{U}_i$ could be viewed as a continuous arbitrary distribution. The general case when $\mathbb{E}(\mathbf{u}_i) = 0$ would be:

$$\mathbb{E}(\bar{D}_i) = \frac{1}{k(k-1)}\mathbb{E}\left(\sum_{j=1}^{k} K(\mathbf{u}_i, \mathbf{u}_{i,j})\right) = \frac{1}{k}\sum_{j=1}^{k} \mathbb{E}(K(\mathbf{u}_i, \mathbf{u}_{i,j}) = \mathbb{E}(\bar{d}_i), \tag{4.6}$$

where $\mathbb{E}(\bar{D}_i)$ is the expectation of $\bar{D}_i$. Then we give a definition of the boundary σ as $\sigma = \frac{\mathbb{E}(\bar{D}_i)}{\mathbb{E}(\bar{d}_i)} = 1$. □

This shows that when the value $F(i)$ reaches 1, we consider it locates at the boundary of clean updates in an FL system without attacks. Thus, to protect the FL learning objective and remove as many malicious remote updates in LoMar, we define the optimal threshold $\epsilon = \min\{1, \epsilon_m\}$. We introduce the development of finding an optimal ϵ and obtaining a trusted joint model $\tilde{\mathbf{w}}^t$ at the t-th iteration in LoMar.

4.3.4 Discussion

In this work, we develop a defense algorithm LoMar to address the poisoning attacks on FL systems. Note that the training data distribution can have a significant influence on both the implementation and defense of poisoning attacks against FL. Typically, as the degree of non-i.i.d training data distribution increases, the learned local updates on the remote clients can be more diverse, which leaves room for the implementation of the attackers and makes it more difficult to develop a defense. Thus, we investigate the performance of the proposed LoMar algorithm under different training data distributions and the results indicate that compared to the existing works, LoMar is more robust under this situation.

We also investigate the performance of the proposed LoMar from the view of FL efficiency. Note that as presented in [107], different from the centralized ML frameworks, the learning speed of an FL network is dominated by the communication cost between the aggregator and remote clients. As such, though the implementation of LoMar defense requires an extra computational cost on the aggregator side, the impact on FL efficiency could be limited.

Moreover, the extra cost of LoMar could be mainly from the development of the k-nearest neighborhood, which is proved as $\mathcal{O}(N)$ in [149]. Typically, in the settings of an FL network, the computational power on the aggregator is much more powerful than remote clients that we believe the linear cost on the aggregator is also limited.

Note that several studies have a close proposal of this chapter, which develop defense approaches to remove malicious updates against poisoning attacks on FL. For example, works in [139] provide Auror, which studies the statistical distribution of remote updates for malicious detection. Meanwhile, [77] provides spectral anomaly detection (SAP) to detect malicious updates for developing a robust FL system. We agree with Auror that though the defender has no access to the training data distribution because of the privacy features, we could approximate it via the observation and investigation of the parameters of the remote updates during the training of the joint model.

However, similar to the existing centralized ML defense approaches, Auror and SAP assume that the defender has access to a publicly trusted dataset, which shares the same distribution with the clean training data. On the contrary, our LoMar uses a non-parametric estimation method KDE, for studying the statistical features without knowing the clean training data distribution, which preserves the important privacy feature of FL. Additionally, the works in [151] investigate the label-flipping attack on FL and provide a PCA analysis-based defense strategy for malicious detection. However, this approach assumes that the malicious and clean updates could be easily divided into two clusters, which is still a global anomaly detection approach that could be easily cheated by model poisoning attacks with a stealth metric [18]. Moreover, we provide the performance evaluation of LoMar against model poisoning attacks in Appendix B.1. In conclusion, compared to recent defense studies on detecting malicious remote clients in FL, our LoMar approach can address the poisoning attacker with stealth metric and develop the defense model under the privacy-preserved FL framework.

4.4 Evaluation

4.4.1 Experiment Setup

To evaluate LoMar, we conduct extensive experiments under the FL framework on multiple real-world datasets. The experiments are performed with Pytorch [117] and we implement the FL framework with the Python threading library by designing remote clients as lightweight threads.

We first introduce four real-world datasets used in this chapter. Especially, to present the imbalanced number of data samples according to different output labels, we assume a reasonable scenario that the training data samples in each remote client could be divided into two parts: one subset of a major label with the most number of training samples and one subset of all output labels with an equal number of samples. Specifically, we use a hyperparameter $\lambda \in [0, 1)$ to represent the ratio of major label training samples in the remote client private dataset.

1) Real-world datasets: We consider four popular datasets in the FL field: MNIST [73], KDDCup99 (network intrusion patterns classifier) [15], Amazon (Amazon product reviews) [6], and VGGFace2 (facial recognition problem from Google image search) [23]. The general information of each dataset is introduced in Table 4.1, which includes the size of the dataset and the number of output labels.

2) Data partitioning settings: In the process of the training data partition, we introduce the hyperparameter λ to control the imbalanced ratio of each remote client. For instance, if the i-th remote client has l_i private training samples, then $(1 - \lambda)l_i$ of the samples are randomly selected from the whole training dataset and the other λl_i samples are chosen from one specific label. For MNIST, we partition the dataset into 1000 remote clients and each remote client has 600 training data samples. Because the KDD dataset has a low number of features and the data samples in each label are imbalanced: some labels only have 20 data samples while others can have over 280,000 data samples. We consider each clean client

Table 4.1: Dataset information overview.

Dataset	Dataset Size	Classes	Features
MNIST	70,000	10	784
KDDCup99	494,020	23	41
VGGFace2	7380	10	150528
Amazon	1500	50	10000

shares the same number of mixed labels and we set the number of remote clients to 23. For the Amazon dataset, because the number of the features is extremely high and the number of data samples per class is very low, we set the number of remote clients to 50 and each client has 150 data samples. And for the VGGFace2 dataset, for simplicity, we use the pre-processed training data before the FL training process (each training image is resized to 256 $\times$ 256). The number of remote clients is set to 200 and the partitioning of the training data follows the same rule in MNIST.

3) Implementation settings: Note that our comprehensive experiments are developed to evaluate the performance of the defense algorithms instead of finding the best learning model with the highest FL learning performance. Based on this motivation, we implement our learning model and we define the default experimental setting of the FL framework as follows: The initialized joint model is set as $\mathbf{w}^0 = 0$. The number of FL iteration time is set as $T = 200$. For each remote training process, the number of remote training is set as $E = 5$. The batch size for each remote training SGD is 20. For MNIST, KDD, and Amazon datasets, which are with less information in each training data sample, we use a logistic regression model with one fully-connected layer. For VGGFace2, we implement a DNN classifier SqueezeNet with the touch vision package [102]. The detailed architecture of our SqueezeNet classifier is described in Table 4.2.

4) Compared defense algorithms: We compare our LoMar defense algorithm with the following existing defense methods: 1) Krum [20] is developed to protect distributed learning models by giving an alternative aggregation rule. At each iteration, this defense computes the Euclidean distance between each update and removes the malicious updates from the

Table 4.2: SqueezeNet model setting.

Layer Type	Size	Parameters	Value
Conv + ReLU	$3 \times 3 \times 64$	Layers	12
Max Pooling	3×3	Batch size	8
Conv + ReLU	7380	Momentum	0.9
Max Pooling	3×3	Weight decay	0.0001
Conv Kernel	10×512	Learning rate	0.001
Output	10		

Table 4.3: Testing accuracy under the label-flipping attack.

	MNIST			KDD			Amazon		
	Overall	Target	Other	Overall	Target	Other	Overall	Target	Other
LoMar	0.912	0.977	0.930	0.971	0.991	0.990	0.970	0.988	0.989
FG	0.783	0.911	0.512	0.901	0.989	0.965	0.922	0.971	0.970
Krum	0.883	0.051	0.815	0.692	0.022	0.970	0.823	0.084	0.915
FG + Krum	0.812	0.996	0.880	0.957	0.985	0.918	0.901	0.960	0.917
Median	0.655	0.925	0.610	0.555	0.971	0.708	0.572	0.959	0.500
No Defense	0.884	0.078	0.925	0.800	0.015	0.998	0.900	0.015	0.998
No Attack	0.931	0.982	0.946	0.980	0.996	0.998	0.990	0.994	0.996

aggregator which has a larger distance. The number of clients removed by the aggregator in our chapter is set to M as the number of selected clients is $N - 0.5 \times M - 2$. 2) FoolsGold(FG) [44] is a defense method against Sybil attacks [37]. It addresses the attacker by penalizing the learning rate of the malicious updates, which are detected by evaluating the angle difference between each update and the joint learning model. We claim that the memory usage in the FG chapter violates the privacy-preserving rule of the FL system and we only take the no-memory version of FG in this chapter. We set the inverse sigmoid function in the FG method centers at 0.5 and the confidence parameter to 1. 3) Median defense method [185] uses an aggregator which sorts the updated value of this parameter among all $N + M$ clients and picks the median value as the contribution to the joint model at each iteration. Note that when $N + M$ is odd, the median value comes from one update, and when $N + M$ is even, it comes from the average of two updates.

In this chapter, we consider two types of poisoning attacks on the FL system: the data poisoning attack (which generates the poisoned remote updates by the malicious data) and

the model poisoning attack (which creates the remote malicious update based on the FL aggregating rule and the attack objective). For the data poisoning attack, we implement the Label flipping attack [19]. And for the model poisoning attack, we implement the stealthy model poisoning method in [18]. Note that in this chapter, the attack methods we select are the most representative and provide a general attacking formulation for different defense algorithms in each category. For instance, [42] provides a typical local model poisoning attack against Byzantine defense mechanisms, however, it only focuses on the Byzantine-related algorithms while not providing a successful attack on other defense strategies. We will introduce the experimental setting and the results of the model poisoning attack in Appendix B.1.

The goal of the label-flipping attack is to take control of the target label(class) in the FL system. Typically, the attacker addresses the target label by flipping the output of target data samples from a source label. In our chapter, we consider the malicious and the clean remote clients to share the same number of training data samples. Specifically, we introduce a parameter $\tau \in [0.1, 1)$ to control the malicious ratio in our experiments. Similar to the definition of λ, for the i-th remote client, the number of randomly selected training data samples is $(1-\tau)l_i$ and the number of label-flipped data samples is τ. We then introduce our default attack implementation on the FL system corresponding to each dataset. For the MNIST dataset, the attacker injects 100 malicious clients into the FL system. The source label is set to digit 7 and the target label is set to digit 1. For the KDD dataset, we pick two classes with more than 200,000 data samples as the source label and target label and create 3 malicious clients whose poisoned samples are flipped from class 11 to 9. For the Amazon dataset, we set the source label to 15 and the target label to 10. The number of malicious clients is set to 5. For the VGGFace2 dataset, we pick up two target labels from two different source labels as label 3 to label 2 and label 10 to label 9. For each target label, we create 10 malicious clients and the total number of malicious clients is 20.

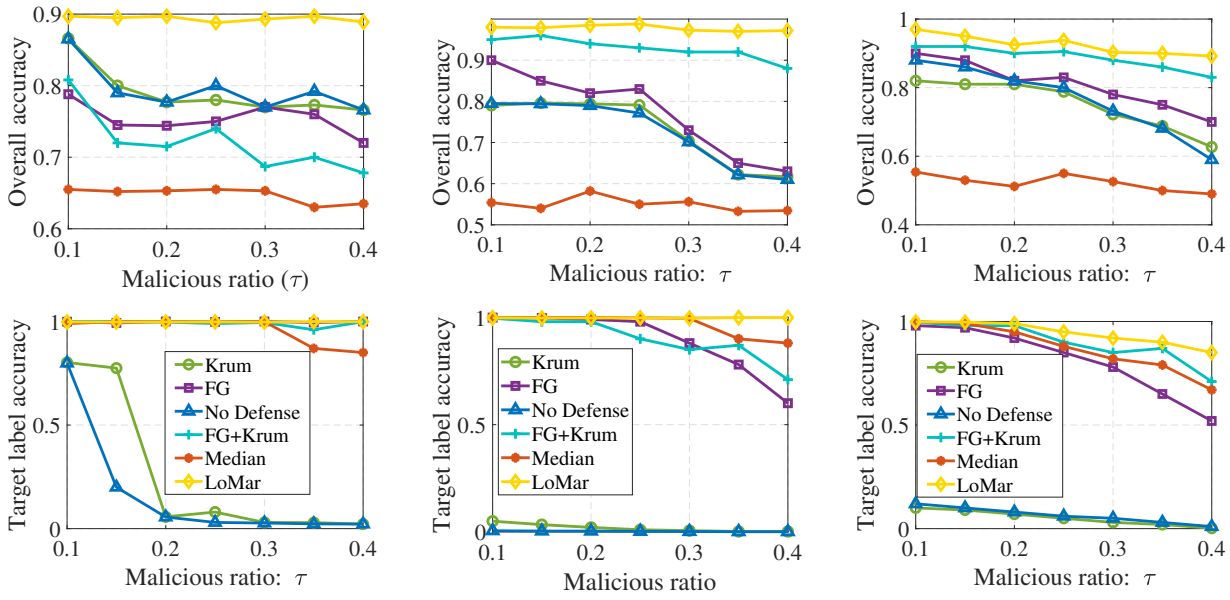


Figure 4.3: Overall and targeted accuracy on the three datasets.

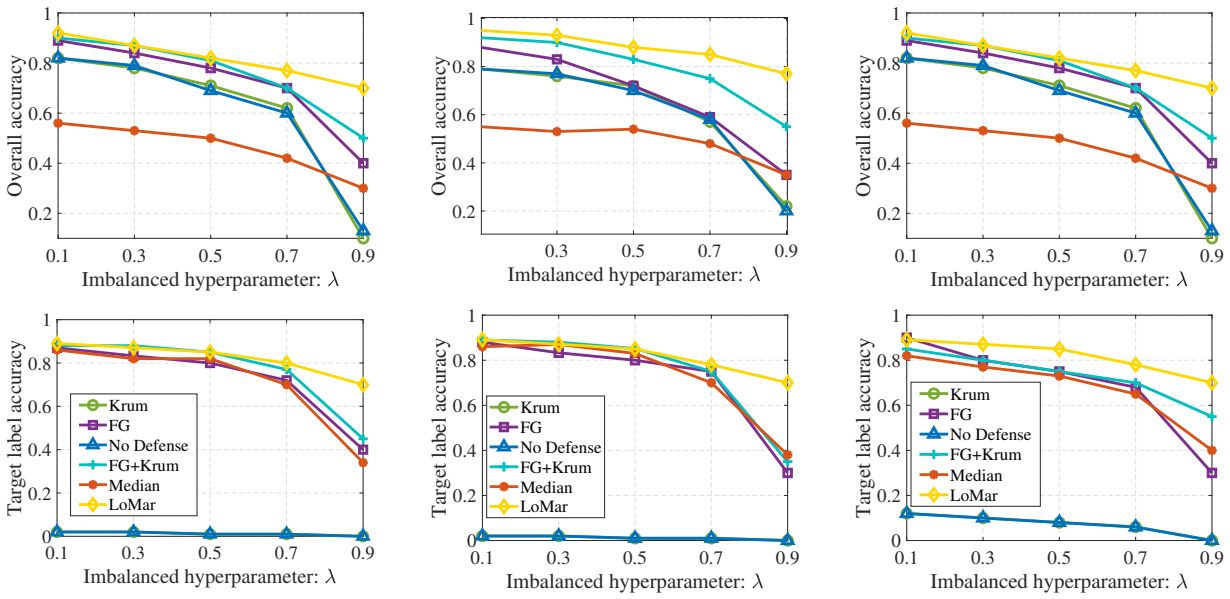


Figure 4.4: Non-iid overall and targeted accuracy on the three datasets.

To evaluate the experiment results accurately, we introduce different evaluation metrics: 1) Target label accuracy. We use the target label accuracy to represent the testing performance of the target labels, e.g., a defense algorithm can be considered to be failed if the target label accuracy is low. 2) Another label accuracy. The other label accuracy indicates the learning result on other labels except for the target label and the source label. For

example, in label flipping $1-7$ attack on MNIST, 1 is the source label while 7 is the target label. If the testing results of other labels are low, we consider the FL system might be harmed by the defender because those labels are not related to the attacker. 3) Overall learning accuracy. We consider the overall learning accuracy denotes the average learning accuracy of all labels in the FL system and it can show the overall learning performance for compared defense methods under poisoning attacks. Note that the results of overall accuracy could be different from the summation of the target and overall accuracy because the performance of the source label could be different due to the attack and defense strategies. 4) Malicious alarm confidence. We also use the receiver-operating characteristic (ROC) [35] to evaluate the malicious alarm confidence of the compared defense algorithms. This metric is introduced in Section 4.4.4.

4.4.2 Results and Analysis

We show the results of our experiments with tables and figures in this section, and we provide a detailed analysis of the results based on our evaluation metrics.

1) Analysis of compared algorithms for each dataset: We give the comparison between LoMar and other defense methods based on three datasets in Table 4.3. We can notice that our LoMar defense model has the best defensive performance among all compared methods on overall accuracy, target label accuracy, and source label accuracy in MNIST and Amazon datasets, for example, LoMar can achieve 97.0%, 98.8%, and 98.9% in Amazon. The result of KDDCup99 dataset shows that LoMar has the second-best learning accuracy on average of all three features, 97.1%, 99.1%, and 99.0%. By taking both attack rate and overall accuracy into consideration, LoMar has the best defensive performance among these defense methods when the percentage of malicious updates increases.

The result of Krum on the three datasets shows that it is similar to the defense scenario so that we can consider Krum cannot defend against label-flipping attacks sufficiently. FG has the better performance of the target label but is worst for overall and other clean labels.

It indicates that FG can influence the clean labels with false malicious detection on clean updates, e.g., 78.3%, 91.1%, and 51.2% for MNIST. Median has the worst performance overall and another label accuracy. Although in all three datasets, it performs sufficiently of the target label, 92.5%, 97.1%, and 95.9%. FG+Krum has the best defensive performance among all three existing defense methods. However, it can still remove the clean updates from FL by mistake, especially in MNIST dataset as the overall learning accuracy is only 81.2%.

2) Analysis of the different malicious ratios τ: Under different malicious ratios τ, we show the performance with the compared defense methods on the MNIST dataset in Figure 4.3. We can see that LoMar has the best performance on overall and target labels under the different values of τ. Even with the number of malicious increasing, LoMar keeps the accuracy as 90% overall and 98% target label. FG and FG+Krum perform sufficiently on the targeted label, but their overall accuracy is worse than no defense scenario. Median defense takes the worst performance as overall accuracy 65%. For Krum, we could find out that the target label accuracy decreases with the increase of τ, even to 2% (similar to no defense).

For KDD in Figure 4.3, LoMar has highest overall and target testing accuracy 98% and 99%. The Median also performs worst for the whole testing 55%. The overall testing accuracy of other defense methods decreases obviously, when τ increases, FG defense decreases from 90% to 63% with the malicious ratio increasing from 0.1 to 0.4. Krum defense cannot defend the target label, which shows as similar to the no defense scenario.

Although the performance of LoMar reduces when more malicious clients are injected for the Amazon dataset in Figure 4.3, it has a better result than other defense methods. Median performs worst for the overall testing 58%, and the target label accuracy reduces from 98% to 62%. Krum performs similar to no defense both on overall and target labels. FG+Krum has overall accuracy 91%-83% and target label accuracy 99%-77%. The overall accuracy of FG is 85%-71% and target label accuracy is 95%-52%.

Table 4.4: Testing accuracy under multi-labels flipping attack.

	Overall Acc.	Acc. 1	Acc. 7	Acc. 6	Acc. 9	Acc. 2	Acc. 5	Acc. 0	Acc. 8	Other Acc.
LoMar	0.885	0.962	0.895	0.936	0.864	0.844	0.786	0.955	0.836	0.875
Krum	0.734	0.834	0.800	0.765	0.750	0.624	0.586	0.750	0.687	0.801
FG	0.802	0.905	0.766	0.918	0.925	0.622	0.457	0.893	0.938	0.766
FG + Krum	0.415	0.859	0.802	0.996	0.991	0.398	0.204	0.040	0.000	0.359
Median	0.655	0.885	0.579	0.812	0.472	0.706	0.282	0.484	0.778	0.619
No Defense	0.705	0.822	0.809	0.763	0.748	0.632	0.585	0.749	0.688	0.771

4.4.3 Analysis of Imbalanced Samples with Different λ

We then evaluate the performance of LoMar with different values of λ, which denotes the ratio of training samples in the major label at each private remote dataset. We setup the range of $\lambda = [0, 1, 0.9]$ and compare the performance of LoMar with the existing defense approaches.

Figure 4.4 shows the overall and target label testing accuracy for different λ values. We could notice from the results that when the value of λ increases, the learning performance could decrease rapidly. We consider this phenomenon occurs because the introduced defense approaches falsely predict the imbalanced clean updates as malicious, especially under a higher imbalance degree. For example, the accuracy of Krum on the MNIST dataset reduces from 61.2% to 15.5%. We also find that though LoMar has a decreased performance at the same time, the decrease degree is lighter when compared to other defense methods. These results show that LoMar still outperforms existing approaches when the remote training samples are imbalanced.

1) Analysis compared algorithms under multiple-labels flipping attack: In this part, we investigate the performance of LoMar under a label-flipping attack with multiple targets. Difference from the previously introduced single target label-flipping attack, the malicious updates in this scenario consists of four different source-target label pairs in the MNIST dataset: 1 to 7, 6 to 9, 2 to 5 and 0 to 8. Note that we choose the source and target labels based on common sense in the development of an attacker: the malicious is more difficult to be found if two labels share a close distribution. Specifically, for better presentation and comparison, we still set the percentage of malicious clients to 10% that each pair controls

25% of malicious updates. In this setting, 4 out of 10 classes are under attack and the detailed experimental results are shown in Table 4.4.

LoMar has the best overall testing accuracy and the accuracy for other labels which are clean 88.5% and 87.5%, and FG+Krum has the worst performance 41.5% and 35.9%, which is lower than no defense scenario 70.1% and 77.1%. For each label, LoMar increases the testing accuracy 10%-20% than no defense, for instance, LoMar is 93.6% for the label 6, and there is only 76.3% without defense. Krum has a similar performance as the no defense scenario, the degree of increase is -0.2%-1.2%. The result indicates that it has limited abilities to defend the attack as the attack rate approaches the upper bound of an attacker in a system without defense. For FG, we can find the defense performance against attacks from label 1 to label 7 and label 0 to label 8 is sufficient, and it fails to other attacks. We assume that the reason FG fails to defend against this attack is that it penalizes the learning rate of many clean updates. For FG combined with the Krum defense model, this method shows extreme results, for example, it has the highest accuracy 99.6% at label 6, but it cannot determine which label is 8, i.e., 0% for label 8. We infer the reason is that the combination of FG and Krum causes more malicious alarms on clean updates which may come from the features of these two defense mechanisms. The Median defense method does not increase the performance sufficiently, but it performs very low for the label 5, i.e., 28.2%. Furthermore, on the hand of learning accuracy, the Median defense fails to obtain a feasible learning accuracy on both target labels and other non-target labels.

2) Evaluation of SqueezeNet for VGGFace2 dataset: We evaluate the defense performance based on the training and testing classification matrix in Table 4.5. Although LoMar defense is influenced by the attacker with 73.3% target label training accuracy and 83.9% of the testing classification. It also has the best accuracy among these defenses. Moreover, it still has the best overall accuracy both on training and testing as 83.0% and 89.7%, and the best training accuracy for other labels 85.4%. We could notice that compared to the FG+ Krum method, LoMar obtains a higher target accuracy and other label accuracy but lower overall

Table 4.5: Learning performance on the VGGFace2 dataset.

Training	Overall Acc.	Target Acc.	Other Acc.
LoMar	0.830	0.733	0.854
FG	0.799	0.679	0.829
Krum	0.834	0.686	0.870
FG+Krum	0.862	0.705	0.846
No defense	0.834	0.653	0.880
Testing	Overall Acc.	Target Acc.	Other Acc.
LoMar	0.897	0.839	0.912
FG	0.867	0.770	0.891
Krum	0.869	0.780	0.891
FG+Krum	0.884	0.831	0.898
No defense	0.872	0.789	0.893

accuracy. We consider this might be because the performance of the source label is enhanced by the byzantine tolerance strategy, which is supported by the results in Krum.

Other defenses do not perform efficiently on the target label either training or testing side, i.e., the accuracy of all defenses is lower than 70% and 80%. For Krum, we find that the performance is only 3% better than with no defense and we consider this method has almost zero ability on dealing with the attackers in this experiment. For FG, it has a weak overall training accuracy of 79.9% and other clean labels 82.9%. This indicates that FG has limited defense performance in addressing attackers at a DNN FL model. In conclusion, the results in DNN models support that the LoMar defense algorithm has the ability to detect attacks and outperforms other methods.

3) Impact of different λ and τ on VGGFace2: The results of impact with different τ on VGGFace2 dataset in Figures 4.5 show that the performance of compared defense methods does not have a significant reduction if more malicious clients are injected into the dataset. It might come from the reason that the defense performance of compared defense methods cannot remove the impact of the attacker completely even from the lowest ratio τ, and attack objectives reach the upper bound as a low malicious update ratio. Figure 4.5 shows that when imbalanced ratio λ increases from 0.1 to 0.7, the performance does not have obvious

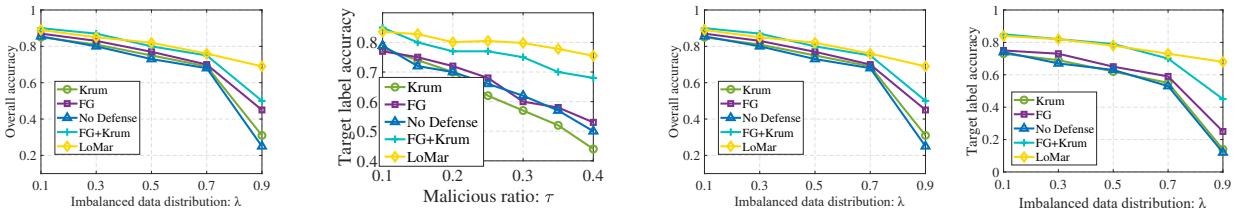


Figure 4.5: Learning performance on VGGFace2.

decreasing. When λ increases to 0.7, we can see existing defense methods reduce sharply, but LoMar also keeps the reduction degree and sufficiently protects the system.

4.4.4 Malicious Alarm Evaluation

In this part, we further investigate the reasons that lead to the different experimental results between our proposed LoMar and compared approaches under FL attacks. Specifically, we study the false alarms in the defense approaches, which could be both the false malicious and the false clean determinations. In this chapter, we use ROC analysis to study this phenomenon, which is a state-of-art statistical model evaluation tool.

For the presentation, we first introduce several important parameters in the ROC analysis. Generally, there are two benchmarks to illustrate the performance of a learned classifier in ML, sensitivity, and specificity. In this work, we denote the number of true clean updates as N and the number of malicious updates as M. Based on N and M, we add four corresponding variables to introduce the definition of remote updates under the compared defense approaches. Specifically, N_f is the number of updates that are falsely determined as clean while N_t is the number of correctly determined clean updates. Meanwhile, M_t represents the number of successfully detected malicious and M_f denotes the number of false alarms. Obviously, we could notice that $N = N_t + M_f$ and $M = N_f + M_t$. As such, the ROC analysis considers the sensitivity as the rate of correctly determined clean updates $\frac{N_t}{N}$ and the specificity as the rate of correctly determined malicious $\frac{M_t}{M}$. In this condition, we could obtain the ROC curve, which is a plot of the sensitivity on the y axis and the values of $1-$ specificity on the x axis. Note that the worst results for a defender classifier in the ROC curve reflect into

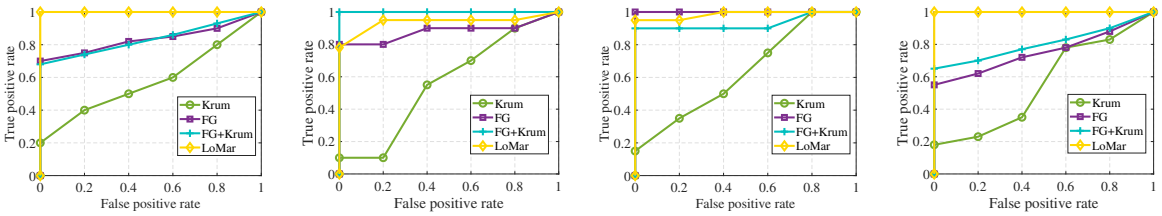


Figure 4.6: ROC on MNIST, KDDCup99, Amazon, and VGGFace2.

a 45° diagonal line from (0,0) to (1,1), which indicates that malicious updates are detected by random chance. And the better the classifier is, the area of the ROC curve is bigger.

Figure 4.6 shows the ROC curves of compared defense methods on datasets mentioned in this chapter. Note that due to the mechanism of Median which obtains the joint model from sorting each parameter of the updates and selecting the median value as the contribution to the joint model, this method leads to a large number of false detecting and we only discuss LoMar, FG, Krum, and FG+krum defense methods.

Firstly, we can notice that Krum has the worst results on each ROC curve. In Figure 4.6, the Krum defense classifier is close to a 45° diagonal line from (0,0) to (1,1) which means that the efficiency of Krum almost approaches a random chance. Secondly, the FG method has the best ROC curve on the Amazon dataset and the FG combined with the Krum method has the best performance on the KDD dataset as the curves are nearly a square area. However, the results of these two defense methods on MNIST and VGGFace2 datasets are not ideal as the areas are far smaller than the results from our LoMar defense method. Especially, our LoMar defense has the best ROC curve on MNIST and VGGface2 datasets and the second performance on KDD and Amazon datasets. Furthermore, the obtained results show that our proposed LoMar has the best overall average ROC curve area on all four datasets, and we believe that the results on ROC are related to defense performance results in Section 4.4.2. The correlation between ROC curves and the FL model performance supports our intuition that compared to existing defense methods, LoMar can develop a successful malicious removing strategy, which provides less number of both the false malicious and the false cleans in the learning process of the FL system.

4.5 Related Works

Poisoning is the most widespread type of attack in the history of the learning field [3, 5, 144, 110]. In general, poisoning attacks reduce the learning model accuracy by manipulating the learning training process to change the decision boundary of the machine learning system. Depending on the goal of poisoning attacks, we classify those attacks into two categories: targeted poisoning attacks [138, 46, 179] and non-targeted poisoning attacks [19, 58, 180, 99]. Non-targeted poisoning attacks are designed to reduce the prediction confidence and mislead the output of the ML system into a class different from the original one [115]. In targeted poisoning attacks, the ML system is forced to output a particular target class designed by the attacker [116]. Compared to non-targeted poisoning attacks, targeted poisoning attacks are more difficult to be found by learning defense systems because targeted poisoning attacks can affect the ML system on the targeted class without changing the output of other classes.

Recent studies [11, 18, 108, 44, 42, 17, 109, 151] on data poisoning attacks explore the privacy and system risks of a decentralized machine learning system. Targeted poisoning data attacks could manipulate the training dataset in the FL system in two ways: label vector manipulation [171] and input matrix manipulation [11]. In label vector manipulation, the attackers can directly modify the labels of the training data into a targeted class, e.g., Label flipping attack [171], where some labels of training data (known as the "target" classes of the attacker) are flipped into another class to reduce the recognition performance of the target classes. Meanwhile, the attackers can also train a generative model for producing poisoning data [82]. On the other hand, the features of training data could be manipulated to achieve the goal of targeted data poisoning attack by input matrix manipulation [18, 42].

One category of existing defense approaches on FL aims to separate the malicious and clean remote clients, e.g., Auror [139]. However, Auror uses the trusted training data to determine a threshold between malicious and non-malicious features, which is not realistic in FL settings. Another type of method is developed based on Byzantine-tolerant learning theory [52, 41, 172, 20, 44]. For instance, the Zeno defense algorithm in [173] removes several

largest descents in each local training iteration and combines the rest of the updates to be the joint model. However, the measurement is based on Euclidean distance, which cannot determine the true largest descents due to the dimension reduction. Trim Mean [83] method achieves the goal by finding a subset of the training dataset to minimize the loss function.

The density-based anomaly detection is widely used in data outlier detection [149, 192, 193]. Several improvements of the basic density-based model have been proposed, for example, a connectivity anomaly detection [150], or based on a reverse nearest neighborhoods adding to the nearest neighborhoods and thinking about the relationship between different values as a measurement of anomaly [137]. In [48], they proposed FIND, a density-based detection to detect nodes with data faults that do not need to assume the sensing model nor the event injections cost. [124] uses density estimation to detect the outlier data in large data streams in online applications.

4.6 Summary

In this chapter, we propose a new two-phase defense algorithm called LoMar to address the poisoning attacks against FL systems. In Phase I, LoMar define a kernel density-based estimator to indicate the degree of the maliciousness for each update compared to the reference set, which is collected by its k-nearest neighborhoods. In Phase II, LoMar designed an asymptotic threshold to provide a binary determination of the poisoned updates. Specifically, the provided threshold also protects the clean updates of the FL system from being regarded as malicious by the defender. Our empirical results on four real-world datasets with the comparison against four existing defense methods demonstrate that LoMar can address both data and model poisoning attacks against FL.

Chapter 5: How to Test the Randomness for Security?

5.1 Abstract

We revisit the traditional framework of wireless secret key generation, where two parties leverage the wireless channel randomness to establish a secret key. The essence of the framework is to quantify channel randomness into bit sequences for key generation. Conducting randomness tests on such bit sequences has been a common practice to provide the confidence to validate whether they are random. Interestingly, despite different settings in the tests, existing studies interpret the results the same: passing tests means that the bit sequences are indeed random.

In this chapter, we investigate how to properly test the wireless channel randomness to ensure enough security strength and key generation efficiency. In particular, we define an adversary model that leverages the imperfect randomness of the wireless channel to search the generated key and create a guideline to set up randomness testing and privacy amplification to eliminate security loss and achieve an efficient key generation rate. We use theoretical analysis and comprehensive experiments to reveal that common practice misuses randomness testing and privacy amplification: (i) no security insurance of key strength, (ii) low efficiency of key generation rate. After revision of our guideline, security loss can be eliminated and the key generation rate can be increased significantly[4].

[4]This paper was published in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3753-3766, June 2021. Permission is included in Appendix D.

5.2 Background and Preliminaries

In this section, we briefly introduce the background of extracting secrets from the wireless channel and then formalize the framework of secret key generation. To this end, we discuss the scenario and assumptions in this chapter.

5.2.1 Extracting Random Secrets from Wireless Channels

Traditional cryptographic mechanisms (e.g., Diffie-Hellman and RSA [63]) reply on establishing computational difficulties for an adversary to achieve the goal of security. In wireless, mobile or IoT domains [198, 169], many wireless security designs have been proposed to leverage the reciprocal and random properties of wireless channel measurements (e.g., RSSI and phase shifts) to generate a common secret sequence between Alice and Bob. In many studies [9, 103, 59, 118, 156, 26, 90, 160, 161, 89, 154, 166, 198, 1, 122, 121, 25], such a sequence is directly used as the secret key for the secure communication between Alice and Bob. In this chapter, we use secret key generation as our main application scenario to study how to test the wireless channel randomness for security, since it is the most representative study of security designs leveraging wireless channel randomness.

Figure 5.1 shows a typical framework for Alice and Bob extracting a common secret sequence from the wireless channel. The framework consists of design components in both the wireless domain and the cryptographic domain. In the wireless domain (shown on the left-hand side of Figure 5.1), Alice and Bob keep measuring the wireless channel response, such as measuring the RSSI, CSI, or phase shifts between them, and then quantify the measurements into bits [9, 103, 59, 161, 90, 198, 195]. Because of the reciprocal property of the wireless channel, their measurements are likely to be the same from the channel between them, and accordingly, their quantified bits sequences should also be likely the same.

Then, Alice and Bob can enter the cryptographic domain [59, 90, 169] as shown on the right-hand side of Figure 5.1 and use information reconciliation [21] and privacy amplification

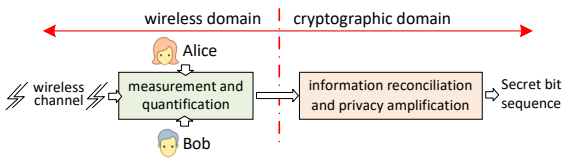


Figure 5.1: A framework of random secret key generation.

[16, 104] to compress their respective bit sequences (with low per-bit entropy) into the same short sequence (with per-bit entropy expected to be near 1).

5.2.2 Formalizing the Framework of Secret Key Generation

For the framework to extract random bit sequences from the wireless channel, there are two major components between Alice and Bob: the wireless domain design and the cryptographic domain design, which are formally modeled in the following.

Definition 3. *(Secret Bit Sequence Extraction Models) The wireless domain design is a function* $f_w : \Omega_D \to \{0,1\}^L$ *mapping a random channel property (e.g., RSSI or phase shifts) in the continuous domain* Ω_D *during a time duration* D *to a binary bit sequence in* $\{0,1\}^L$*, which denotes the set of all bit sequences with length* L*.*

The cryptographic domain design is a function $f_c : \{0,1\}^L \to \{0,1\}^M$ *mapping a binary bit sequence with length* L *to a new sequence with shorter length* $M \leq L$*, in which the correlation among bits is minimized close to 0 by privacy amplification. When* $M = L$*, there is no cryptographic domain in a secret key generation design [103, 1], we simply set function* f_c *as* $f_c(x) = x$ *for any input* x*.*

A statistical randomness test is a function $T : \{0,1\}^* \to \{\{Accept\ H_0\}, \{Accept\ H_1\}\}$*, where* $\{0,1\}^*$ *denotes the set of bit sequences with any length (e.g., length of* L *or* M*),* H_0 *and* H_1 *are null and alternative hypotheses denoting the events that the randomness test succeeds and fails, respectively.*

The objective of Alice and Bob is to leverage the random channel property between them, denoted by $\omega_D \in \Omega_D$[5] during time period D, to compute a bit sequence $K_D = f_c(f_w(\omega_D)) \in \{0,1\}^M$ for their security design purpose.

In the extraction models, there is no evaluation that the bit sequence K_D is sufficient for security purposes. Therefore, security evaluation is another critical component for any wireless channel randomness-based security design. To this end, NIST statistical randomness test suite [14] is widely adopted as a common practice in the literature [103, 59, 161, 131] to test whether the generated bit sequence is random for the security purpose.

5.2.3 Testing Randomness from Wireless Channels

The procedure of a randomness test in the NIST test suite [14] to test the bit sequences extracted from the wireless channel is straightforward: for a bit sequence X quantified from the wireless channel, compute the statistics of X based on a particular test, called P-value, and compare this P-value with a threshold α. The test succeeds if the P-value is greater than α, and fails otherwise.

For Alice and Bob, failing the NIST tests indicates that the wireless channel measurement does not have enough randomness [59]. They have to wait for a better channel condition or adjust their design parameters and then test again. Thus, randomness tests serve a critical role in evaluating the security of a design leveraging wireless channel randomness.

Firstly, it seems perfectly fine to use randomness tests for security evaluation because they are recommended for cryptographic use. But the key question here is not why, but how to use them to test the wireless channel randomness for security? We notice that existing studies adopt statistical randomness tests in different ways for security evaluations. Particularly, two major discrepancies exist in the literature. 1) Where to set the randomness test? There indeed exists a discrepancy in the literature to place the randomness test: a large number of

[5]Alice and Bob may not observe exactly the same ω_D in practice because of noise and interference. In this regard, denote by ω_D^A and ω_D^B Alice's and Bob's observations respectively. Robust wireless domain design aims to achieve $f_w(\omega_D^A) = f_w(\omega_D^B)$ and information reconciliation also ensures $f_c(f_w(\omega_D^A)) = f_c(f_w(\omega_D^B))$. So $\omega_D^A \neq \omega_T^B$ does not affect our security analysis. For the sake of simple notation, we let $\omega_T = \omega_T^A = \omega_T^B$.

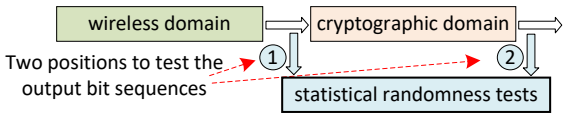


Figure 5.2: The use of statistical testing in secret key generation.

designs [59, 122, 90, 89] choose to test the bit sequences at Position 2, and other designs test at Position 1 [103, 1, 26]. 2) How to choose a critical parameter, the P-value threshold α, in randomness tests? The value of α represents the confidence level of the test output. We notice that $\alpha = 1\%$ is dominantly adopted in existing studies [103, 1, 59, 122, 90] according to the NIST test suite [14]. However, some studies choose $\alpha = 5\%$ for the tests, for example, [169] tests 200-bit sequences generated from the wireless channel between mobile devices.

These observations reveal that despite the advance in efficiently quantifying the wireless channel randomness (e.g., RSSI or CSI) into bit sequences, the common practice of using randomness testing exhibits discrepant setups for security evaluation. Are these setups equally secure, or secure enough for a particular application? As a result, the focus of this chapter is to design a rigorous mechanism to understand how to properly use randomness tests for security evaluation of these security designs quantifying wireless channel randomness.

5.3 Problem Formulation and Research Statement

We first formalize the models in extracting secret bit sequences from the wireless channel, then identify research challenges and propose the guideline for setting up statistical randomness tests.

5.3.1 Formalizing Statistical Randomness Testing for Security

It is clear in Definition 3 that extracting a secret bit sequence does not rely on the statistical randomness test T. The role of T is to ensure security by denial: if the bit sequence from the channel fails the test T, then the channel randomness is not sufficient for security purpose.

However, a randomness test can be set up in many different ways (e.g., varying setups observed in the literature: [103, 59, 90] vs [26, 169]). Since a randomness test only considers some specific parts to evaluate the degree of randomness for a bit sequence, the bit sequence can always pass a randomness test as long as its construction is biased toward the test. Blindly setting up the randomness tests provides no guarantee of security. We have to rethink how to quantify the extent to which the security by denial via a randomness test meets the security goal of Alice and Bob, i.e., obtaining a secret bit sequence K_D of length M.

To this end, we need to design T in two steps: (1) Designing a technical adversary model Eve against Alice and Bob. However, there is no such model proposed in the literature, which makes a formal analysis for wireless randomness-based security difficult. (2) Defining Eve's attack success probability as a function of the randomness test. In this way, we can quantitatively measure the benefit of security by denial via a randomness test and properly set the test.

If the two steps are in place, we are able to evaluate whether a randomness test is properly set up for protecting the system security. Specifically, we aim to compare Eve's strategy under the randomness test with the benchmark random guess (RG) strategy in terms of the success probability, and set up the randomness test such that

$$\mathbb{P}(\text{Eve succeeds}) \leq \mathbb{P}(\text{RG succeeds}). \tag{5.1}$$

In other words, we must set up the randomness test such that Eve's attack is no better than RG. We also define the security loss due to the randomness testing as

$$L_{\text{security}} = \log_2(\mathbb{P}(\text{Eve succeeds})/\mathbb{P}(\text{RG succeeds})). \tag{5.2}$$

For example, if $\mathbb{P}(\text{Eve succeeds}) = 2^{-80}$ and $\mathbb{P}(\text{RG succeeds}) = 2^{-120}$, the security loss L_{security} is computed as 40, which is the difference between the exponents in the two probabilities. In general, the security loss L_{security} can have a negative value for some naive attack strategy

(e.g., Eve always tries a fixed guess every time, which is even worse than RG). In this chapter, we only consider non-trivial cases with $L_{\text{security}} \geq 0$ (i.e., a security loss is non-negative).

5.3.2 Formalizing Statistical Randomness Testing for Efficiency

In Section 5.3.1, we formalize the role of statistical randomness testing from the security perspective. It is also worth noting that efficiency for secret key generation is another important factor to consider. Note that the efficiency comes from two aspects: randomness test T (i.e., the probability of bit sequence can be accepted or rejected) and privacy amplification rate R_{privacy} (i.e., the compressed rate M/L). If the statistical randomness test T is set too strict, it will be difficult to generate wireless secrets during a short period because T rejects the channel samples too many times. On the other hand, privacy amplification allows the input of a correlated bit sequence and compresses it into a short sequence with higher per-bit entropy. A higher R_{privacy} can lose the design of the test T, but at the same time reduce the efficiency because more bits are compressed, which indicates more bits extracted from a wireless channel are discarded. In this chapter, we only aim to achieve high efficiency by controlling T and R_{privacy}. Thus, we consider the efficiency E as our evaluation and formally define it as

$$E = \mathbb{P}(T \text{ accepts } H_0) \cdot R_{\text{privacy}}, \tag{5.3}$$

and the efficiency loss is defined as $L_{\text{efficiency}} = 1 - E$. Note that there are two situations of placing randomness test T in the literature: if we place randomness test T at position 1, $\mathbb{P}(T \text{ accepts } H_0) = \mathbb{P}(T(f_w(\omega_D)) \text{ accepts } H_0)$. If T is at position 2, $\mathbb{P}(T \text{ accepts } H_0) = \mathbb{P}(T(K_D) \text{ accepts } H_0)$. For simplicity, we generalize these two situations into one formula $\mathbb{P}(T \text{ accepts } H_0)$. With security loss L_{security} and efficiency loss $L_{\text{efficiency}}$, we can quantitatively evaluate a wireless secret bit generation design in terms of its security and efficiency. To properly set up the randomness testing, we must ensure that its security loss L_{security} is zero under an adversary model and at the same time we also need to maximize its efficiency E or, equivalently, minimize the efficiency loss $L_{\text{efficiency}}$.

5.4 Formal Adversary Model

We consider a wireless channel randomness-based design scenario shown in Figure 5.2. We assume that all design specifications and parameters in the wireless domain (e.g., bandwidth, carrier frequency, and quantization methods) and the cryptographic domain (e.g., cryptographic methods) in Figure 5.2 are known to the public. An adversary Eve can hear all communications between Alice and Bob, but cannot access Alice's or Bob's antenna. Therefore, she cannot directly measure the accurate channel response between Alice and Bob. We assume that Eve can neither actively affect the wireless channel between Alice and Bob, nor modify the content of any communication.

We also assume that Eve has a powerful yet finite computational capability. This enables Eve to perform intensive computations, leveraging the imperfect randomness of the wireless channel measurements, to search for the secret between Alice and Bob. Such an assumption of Eve's practical computational capability helps offer intuitive measurements of security degradation to indicate the importance of correctly setting up statistical testing for wireless security. For example, the 2017 SHA-1 collision attack has an estimated computational effort equivalent to $2^{63.1}$ SHA-1 calls [145]. We define Eve's capability and objective as follows.

Definition 4. *(Eve's Capability and Objective) Given Definition 3, Eve aims to develop a key search strategy to maximize her success probability by performing N searches for the secret K_D, and N is called Eve's capability.*

5.4.1 Secrets Generated from Wireless Channel Randomness

Given the fact that all models in Definition 3 are publicly known, the objective of Eve is to develop a strategy to efficiently search for K_D without exact knowledge of Alice and Bob's channel response ω_D. Existing studies have well explored the building of functions f_w and f_c to obtain a key, but never fully investigated Eve's strategy. Suppose Eve has no smarter

strategy but RG, if we assume that she has a maximum capability 2^{64}, the probability is $2^{64}/2^{128} = 2^{-64}$ to obtain a 128-bit key generated by Alice and Bob.

Is there a smarter strategy for Eve to do better? We first analyze how the wireless channel generates secret bit sequences. [103] proposed the basic idea of the level-crossing algorithm: the channel information (e.g., RSSI or CSI) is estimated over a time interval (TI) larger than the coherence time, set two thresholds q_+ and q_- by calculating magnitude or phase; the measured channel information will be quantified to 1, if it is greater than a threshold q_+ or 0 less than q_-. It is widely suggested in existing works [103, 59, 198] that the measurement interval should at least equal the channel coherence time such that the measured samples are considered approximately independent.

Nonetheless, this approximate independence assumption creates a very vague boundary from the security perspective: the measurements from the wireless channel are correlated [103, 59, 198, 64]; although the correlation becomes weaker and could be considered approximately independent for traditional performance analysis when the measurement interval increases, it still makes sense for security analysis to assume that the output bits from the wireless domain are statistically correlated rather than approximately independent. In other words, the input of function f_w is regarded as a correlated signal, leading to a correlated output model of f_w.

Definition 5. *(Wireless Bit Generation Model) Given a channel measurement and quantification period D, the output from the wireless domain, denoted as bit sequence $X = f_w(\omega_D) = [x_1, x_2, \cdots, x_L]$, is modeled as a binary correlated sequence with correlation coefficient $\rho \in [-1, 1]$ for consecutive bits x_i and x_{i+1} for $i \in [1, L-1]$, which is written as $\rho = \frac{\text{cov}(x_i, x_{i+1})}{\sigma(x_i)\sigma(x_{i+1})}$, where $cov(x_i, x_{i+1}) = \mathbb{E}\left((x_i - \mathbb{E}(x_i))(x_{i+1} - \mathbb{E}(x_{x+1}))\right)$ is the covariance between x_i and x_{i+1}, and $\sigma(x_i)^2 = \mathbb{E}((x_i - \mathbb{E}(x_i))^2)$ is the standard deviation of x_i.*

Definition 5 offers a more practical and generic model compared with the traditional one that assumes that channel samples are approximately independent with the sampling duration larger than the coherence time used in the literature. Apparently, we can set

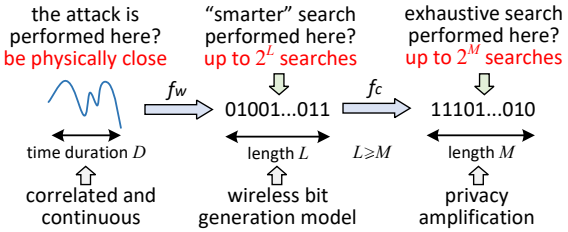


Figure 5.3: Eve's perspective on the secret key generation.

$\rho = 0$ to obtain from the correlated model to the traditional one. Moreover, from a security perspective, we should always assume a defective (rather than perfect) randomness model for security design. A good wireless domain design should generate a bit sequence with a correlation coefficient ρ close to 0. But we can never know if a design indeed achieves 0 in practice. Thus, it is always good to assume $|\rho| > 0$ even if it is a very small value. Accordingly, Eve can leverage such a model to construct her attack strategy, which in turn facilitates formal security analysis for statistically testing wireless channel randomness.

5.4.2 Eve's Strategy

Given the bit generation model from the wireless domain, let us look at the secret bit generation from Eve's perspective, shown in Figure 5.3. As Eve knows $K_D = f_c(f_w(\omega_D))$, there are three straightforward strategies.

1) Search for the secret K_D in the $\{0,1\}^M$ space. There is no evident strategy better than RG because the last step of f_c is privacy amplification.

2) Search for the wireless domain output $X = f_w(\omega_D)$ in the $\{0,1\}^L$ space. Then, compute $K_D = f_c(X)$ because f_c is public. Note that $L \geq M$, so is it really worth searching in a potentially larger space? We find that leveraging the bit correlation to search for X in $\{0,1\}^L$ can result in a better success probability than RG in $\{0,1\}^M$.

3) Search for ω_D, then compute $K_D = f_c(f_w(\omega_D))$. We note that this is possible only if Eve can physically access Alice's or Bob's antenna. We assume that Eve has no such access.

In the three strategies, we show that the second one for Eve (i.e., searching for $X = f_w(\omega_D)$ then computing $K_D = f_c(X)$) can generate a higher success probability if Eve leverages the

correlation in the wireless bit generation model in Definition 5. Figure 5.4 illustrates an example of how the first 4 bits x_1, x_2, x_3 and x_4 from X are generated: the wireless channel is slowly varying and the wireless domain design samples and quantifies the channel response into bits in a sequential way. The first two bits x_1 and x_2 are 1 and the channel changes so the last two bits x_3 and x_4 are 0.

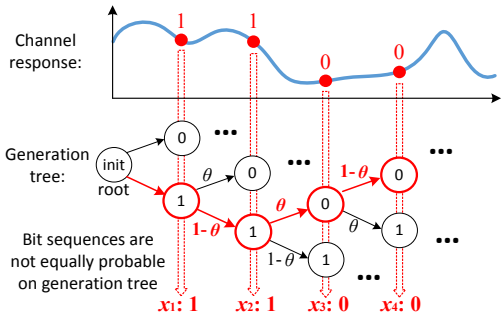


Figure 5.4: Bit generations from the wireless domain forms a generation tree.

According to Definition 5, we map the wireless-domain generation into an abstract process in a generation tree as shown in Figure 5.4, which enumerates all possible bit values generated sequentially. A path from the root to a leaf can represent a generated bit sequence. For example, Figure 5.4 shows the path that generates 1100. We denote by θ the value transition probability in the tree (i.e., the probability that the values of two consecutive bits are different). When the correlation coefficient is larger than 0, the correlation among bits means that a generated bit is more likely to have the same value as the previously generated bit, i.e., θ should be smaller than 0.5. If the correlation coefficient is smaller than 0, θ should be larger than 0.5, respectively. As a consequence, all paths from the root to the leaves in the tree exhibit different probabilities. This helps Eve because she can search for X by starting from the most likely bit sequence towards the least likely bit sequence in the tree. We call such a strategy maximum-likelihood tree search (MLTS).

MLTS maximizes Eve's success probability if bits in X are statistically correlated (i.e. $|\rho| > 0$), and has equal performance to RG otherwise (e.g., $\rho = 0$). In the following, we show the attack performance of MLTS.

Theorem 10. *(Maximum-Likelihood Tree Search) For the sake of simple notation, we let Eve's computational capability N in Definition 4 satisfy $N = \sum_{i=0}^{n/2} \binom{L}{i} + \sum_{j=0}^{n/2} \binom{L}{j}$, where $0 \leq n \leq L$. Then, given the fact that a secret K_D has been established, the attack success probability of MLTS is*

$$\begin{aligned} &\mathbb{P}(\text{MLTS succeeds} \,|\, K_D \text{ established}) \\ &= I_{\frac{1-\rho}{2}}(L - \frac{n}{2}, \frac{n}{2} + 1) + I_{\frac{1+\rho}{2}}(L - \frac{n}{2}, \frac{n}{2} + 1) = I_{MLTS}, \end{aligned} \tag{5.4}$$

where $I_x(a, b)$ is the regularized incomplete beta function$I_x(a, b) = \frac{B(x;a,b)}{B(a,b)}$ with incomplete beta function $B(x; a, b) = \int_0^x t^{a-1}(1-t)^{b-1}dt$ and complete beta function $B(a, b) = B(1; a, b)$.

Proof: See Appendix C.1 for details. □

The advantage of MLTS is that it does not need to know the value of ρ and the transition probability θ to work. Eve should always try to use MLTS in practice to search for X and then compute K_D. It is worth noting that we use the number of searches as an indicator of computational complexity. We consider Eve performs one search on a sequence if Eve spends some computations on the sequence. Due to the use of randomness testing and the use of hundreds of bits as a key in today's practice, Eve cannot easily exclude a wide range of sequences of hundreds of bits (or easily prone to a large branch of the search tree) during searching for the correct key. Eve has to test sequences one by one. Even for a bit sequence that fails the randomness test, she still has to test it (thereby spending some computational time) before knowing that it cannot be used as the key. Or Eve can skip the test and directly spend computations on verifying if a key candidate is correct. These computations on the bit sequence constitute one search even though the bit sequence is test-compliant or not. Therefore, there is no straightforward way to skip all the sequences that are not test-compliant. Knowing the fact that K_D is established does not reduce the number of searches to be performed by Eve. Note that we do not consider trivial cases here (e.g., Eve can simply exclude all 0 or 1 bits).

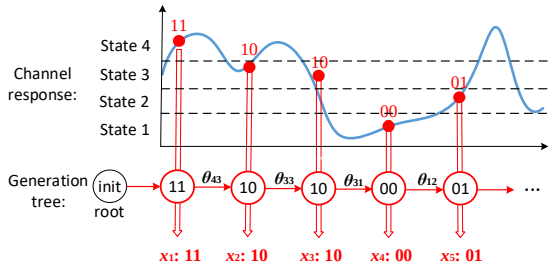


Figure 5.5: The secret key generation of 4-level quantization.

To improve the efficiency of key generation, multi-level quantization methods have been developed in [184, 187]. The core of MLTS, which is to search from the most likely sequence to the least likely sequence, can also be applied to multiple-level quantization. Fig. 5.5 shows an example of 4-level quantization of the channel response. The quantization includes 4 states (i.e., 11, 10, 01, 00). As the figure shows, the system first quantifies the channel response to state 4 (11), then state 3 (10), state 3 (10), state 1 (00), and state 2 (01), which leads to the sequence of 1110100001. The correlation between the two states is stronger if they are closer. As a result, Eve's MLTS can be executed from the most likely sequence to the least likely sequence on a tree with multiple-bit states (instead of single-bit ones) as individual nodes. In this chapter, we will focus on single-bit quantization as it has been widely used in existing studies [103, 59, 156, 26, 90, 122, 169]. We provide a basic performance analysis of MLTS for multiple-level quantization in Appendix C.2.

5.5 Guidelines for Statistical Randomness Test Settings

With the clearly defined MLTS-based attack model for Eve, we are ready to address how Alice and Bob should test the wireless channel randomness for security. Alice and Bob must make sure that they will not create a common secret from the wireless channel with a high correlation over time. On the other hand, they never know the exact value of channel correlation in practice. Then, it seems natural for them to test the channel first, and then make a binary decision, which formalizes the role of the randomness test T in Definition 3.

In this section, we analyze Eve's success probability as a function of randomness testing, then propose the guideline for Alice and Bob to properly set up the randomness testing for security and efficiency.

5.5.1 Eve's Success Probability of Randomness Testing

Randomness testing aims to eliminate the security loss defined in (5.2) by denial and we should make sure that Eve's success probability is no better than the RG's success probability (i.e., no security loss). Based on Definition 3, we define $\mathbb{P}(\text{Eve succeeds})$ as

$$\mathbb{P}(\text{Eve succeeds}) = \mathbb{P}(T \text{ Accept } H_0)\mathbb{P}(\text{Eve succeeds}|H_0) \leq \mathbb{P}(\text{RG succeeds}), \tag{5.5}$$

where $\mathbb{P}(\text{Accept } H_0)$ is the probability that randomness test T passes, which depends on the settings of T. $\mathbb{P}(\text{RG succeeds}) = N2^{-M}$ with Eve's capability N and key length M. $\mathbb{P}(\text{Eve succeeds}|H_0) \approx I_{\text{MLTS}}$ obtained in (5.4), denoting the probability that Eve obtains the key Ω_D conditioned on randomness test T passes[6].

According to our analysis, Eve should always use MLTS in practice and hope for a large $|\rho|$. As a result, Eve's success probability can be written as

$$\begin{aligned}\mathbb{P}(\text{Eve succeeds}) &= \mathbb{P}(\text{MLTS succeeds} \,|\, K_D \text{ established})\mathbb{P}(K_D \text{ established}) \\ &= I_{\text{MLTS}} \cdot \mathbb{P}(T \text{ accepts } H_0) \leq \mathbb{P}(\text{RG succeeds}),\end{aligned} \tag{5.6}$$

where the last equality follows from Theorem 10. Due to the fact that the binary sequence can be considered random enough after privacy amplification, MLTS only can focus on the sequence in the wireless domain. The more random (randomness test can reject large $|\rho|$ scenarios) and longer binary sequence ($L \geq M$) could eliminate the security loss.

[6]Eve aims to search the wireless domain output X yielding the key K_D. However, due to hash collision in privacy amplification, there exists a probability $\mathbb{P}(\text{collision})$ that Eve finds another bit sequence $X' \neq X$ satisfying $K_D = f_c(X) = f_c(X')$. As a result, $\mathbb{P}(\text{Eve succeeds}) = I_{\text{MLTS}} + \mathbb{P}(\text{collision})$, where $\mathbb{P}(\text{collision})$ can be approximated as $1 - (1 - 2^{-M})^L$ in [51]. Since privacy amplification f_c is always designed to make the collision probability $\mathbb{P}(\text{collision})$ negligible, for example, $L = M = 32$ and $N = 16$, $I_{\text{MLTS}} \geq \mathbb{P}(\text{RG}) = 1.53 \times 10^{-5}$ and $\mathbb{P}(\text{collision}) = 7.45 \times 10^{-9}$ such that $I_{\text{MLTS}} \gg \mathbb{P}(\text{collision})$. Therefore, we approximate that $\mathbb{P}(\text{Eve succeeds}) \approx I_{\text{MLTS}}$ for a sufficiently large capability N for Eve in this chapter.

5.5.2 Observations and Design Guideline

From (5.6), we can only guarantee that there is no security loss but the test T might be set too strict to cause a lower efficiency E. Hence, efficiency E is also important and should be considered at the same time as the design guideline. As a result, the design guideline is proposed to find the settings for test T and the privacy amplification rate R_{privacy} to maximize efficiency under the constraint of no security loss, which is written as follows

$$\max \; E = \mathbb{P}(T \text{ accepts } H_0) \cdot R_{\text{privacy}} \tag{5.7a}$$

$$\text{s.t.} \quad I_{\text{MLTS}} \cdot \mathbb{P}(T \text{ accepts } H_0) \leq \mathbb{P}(\text{RG succeeds}). \tag{5.7b}$$

The design guideline aims to find the settings for test T and the privacy amplification rate R_{privacy} to maximize efficiency under the constraint of no security loss. (5.7b) ensures that $L_{\text{security}} = 0$ by selecting the proper P-value threshold α and R_{privacy}. We provide the theoretical analysis of how to calculate $\mathbb{P}(T \text{ accepts } H_0)$ for different α values under different randomness tests in Appendix C.3. Although we have both theoretical results of I_{MLTS} and $\mathbb{P}(T \text{ accepts } H_0)$, there is no straightforward convex or concave property (i.e., increasing α and decreasing R_{privacy} may both satisfy (5.7b)). In practical systems, α and R_{privacy} have typical value ranges and we select $\alpha \in [0.0001, 0.3]$ and $R_{\text{privacy}} \in [0.1, 1]$ in this chapter. Within the ranges, we use the greedy search with small granularity to find the best pair that maximizes (5.7a). From the design guideline (5.7), we can answer the questions in Section 5.2.3.

1) The cryptographic domain function f_c is based on privacy amplification, however, over-estimating the entropy cannot be avoided, since it is generally difficult to accurately estimate the per-bit entropy of a physical source [155]. Consequently, if the randomness test T is set in the cryptographic domain, it is equivalent to testing the output of a sufficiently random sequence, and always passing the test T. Therefore, it is reasonable to test the wireless domain output $X = f_w(\omega_D)$ when extracting randomness from the wireless channel.

2) Based on (5.7), we can solve the optimization function to find the sufficient P-value threshold α and R_{privacy}, instead of manually setting the parameters, to ensure no additional loss in security and achieve high efficiency.

Randomness test T is an important part of the guideline (5.7). Rather than designing a new randomness test, we focus on NIST randomness tests as they have been well structured and widely adopted [103, 59, 156, 26, 90, 160, 161, 89, 154, 198, 1, 122, 169]. In order to configure the NIST randomness tests, we need to analyze the relationship between $\mathbb{P}(T \text{ accepts } H_0)$ and (ρ, α) for a specific test. Hence, in the following, we present how to bridge $\mathbb{P}(T \text{ accepts } H_0)$ to $h(\rho, \alpha)$, where $h(\cdot)$ represents the probability function of (ρ, α) for a specific NIST test.

In many scenarios, multiple randomness tests can be used together for testing. This combination can enhance security and indeed loosen the R_{privacy} setting. However, the setup for a single test in existing studies is not loosened even when multiple tests are used. This has been common in existing studies for wireless key generation [103, 59, 156, 26, 90, 160, 161, 89, 154, 198, 1, 122, 169], hardware security [120], cryptography and software security [100, 71]. This is due to two major reasons: i) it can be mathematically intractable to analyze the joint correlation among multiple tests. The NIST guideline [14] performed such a correlation study and only shows that empirically the correction among NIST tests is very small. As a result, it can be difficult to show how much the setup for each test can be loosened analytically. ii) Using a single-test setup can ensure the worst-case security guarantee even when multiple tests are used. We also adopt this practice in the chapter and recommend the use of the single-test setup for multiple-test scenarios.

5.5.3 Analysis of NIST Randomness Tests

There are fifteen tests provided in the NIST test suite [14], and we choose nine of them which are commonly used in the existing literature [103, 59, 156, 26, 90, 160, 161, 89, 154, 198, 1, 122, 169]. The nine tests in our study are frequency test, frequency test within a block,

runs test, test for the longest run of ones in a block, discrete fourier transform test, non-overlapping template matching test, approximate entropy test and serial test (serial test has two orders). Based on the P-value computation formula, these tests can be categorized into two classes: Gaussian and chi-square distribution. In the following, we use the most common frequency test T_{freq} as the representative to show the procedure of the relationship between $\mathbb{P}(T_{\text{freq}}$ accepts $H_0)$ and (ρ, α). The results of other tests are provided in Appendix C.3. We use the function $h_{\text{freq}}(\rho, \alpha)$ to represent frequency test in NIST. Given a bit sequence $X = [x_1, x_2, ... x_L]$ from Definition 5,

$$h_{\text{freq}}(\rho, \alpha) = \mathbb{P}(T_{\text{freq}} \text{ accepts } H_0) = \mathbb{P}(|S_{\text{freq}}(X)| < \sqrt{2}\text{erfc}^{-1}(\alpha)), \tag{5.8}$$

where $S_{\text{freq}}(X) = \frac{1}{\sqrt{L}} \sum_{l=1}^{L} (2x_l - 1)$ is the statistics definition of frequency test. Since the correlated sequence X can be considered as generating from a uniformly ergodic Markov chain [60], $S_{\text{freq}}(X)$ can be derived by the Markov central limit theorem, only if we know the mean and variance. $|S_{\text{freq}}(X)| \sim \mathcal{N}\left(0, \frac{1+|\rho|}{1-|\rho|}\right)$ is followed by Gaussian distribution. Thus, we have the $h_{\text{freq}}(\rho, \alpha)$ as follows

$$h_{\text{freq}}(\rho, \alpha) = \text{erf}\left(\text{erfc}^{-1}(\alpha)\sqrt{\frac{1-\rho}{1+\rho}}\right), \tag{5.9}$$

where erf and erfc^{-1} are error function and inverse complementary error function. Based on the analysis of randomness test T, we can choose the proper pairs of P-value threshold α and privacy amplification rate R_{privacy}.

5.6 Experimental Evaluation

In this section, we obtain the security loss, efficiency loss, and bits mismatch rate of the secret key generation by the wireless channel response. In the following, we first introduce the experimental setup. Then, we compare the performance of existing secret key generation methods before and after being revised by our design guideline in (5.7) under different experimental environments, different randomness tests, and different lengths of keys.

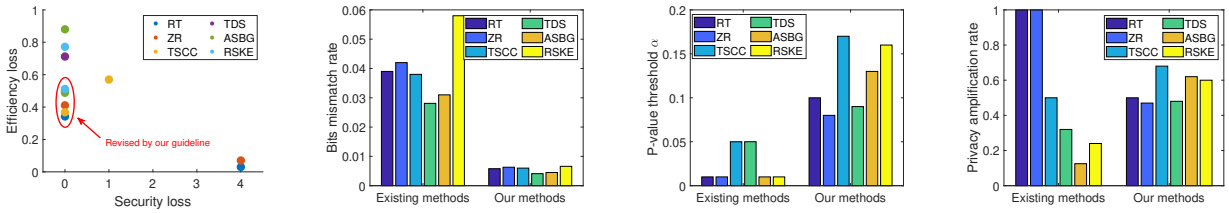
5.6.1 Experimental Setup

1) Channel response measurements: The first step towards analyzing the randomness of the secret key generation in the wireless domain is to collect a large number of channel information (RSSI, CSI, and Phase shifts) in realistic environments. We use two USRP X310s, acting as a transmitter and a receiver respectively, to build our experimental platform, where each device is equipped with a UBX-160 daughterboard and a VERT 2450 antenna. The software toolkit is GNURadio. We implement a typical training data-aided time and frequency synchronization scheme based on [136] for channel probing whose procedure follows [103]. For the equalization, we adopt a frequency-domain OFDM equalizer with the aid of pilot tones [57]. To measure the channel information, the transmitter consistently sends training sequences (known as the preamble in wireless standards) to the receiver with fixed transmit power. On this experiment platform, we collect more than 1 billion channel information in total spanning over 24 hours in different environments with 2.4GHz carrier frequency and 1.0MHz bandwidth.

Table 5.1: Parameters setting of existing methods.

Examples	Test setting domain	α value	R_{Privacy}	Source
RT	wireless	0.01	1	RSSI
ZR	wireless	0.01	1	RSSI
TSCC	wireless	0.05	0.5	Phase
TDS	cryptographic	0.05	0.32	CSI
ASBG	cryptographic	0.01	0.125	RSSI
RSKE	cryptographic	0.01	0.24	RSSI

2) Secret key generation model: We compare the performance of 6 existing secret key generation models: Radio-telepathy (RT) [103], Zero Reconciliation (ZR) [1], Temporally and Spatially Correlated Coefficients (TSCC) [26], The Dancing Signals (TDS) [169], Adaptive Secret Bit Generation (ASBG) [59] and Robust Secret Key extraction (RSKE) [122], where the P-value threshold α, R_{privacy}, different NIST statistical randomness test setting position and source of secret key generation are shown in Table 5.1.



(a) Security loss vs efficiency loss

(b) Bits mismatch rate

(c) P-value threshold α

(d) Privacy amplification rate

Figure 5.6: Basic experimental performance.

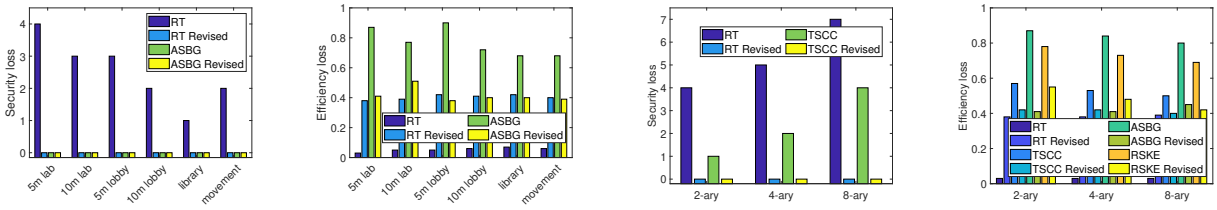
3) Eve's capability: The attacker Eve aims to obtain the secret key K_D through MLTS without knowing any channel information. Although a realistically powerful capability for Eve is $2^{63.1}$ [145], it is still statistically insignificant in the experimentation in order to crack a long key, for example, the attack success probability of cracking a 128-bit key is about 2^{-64}. Thus, we consider a more powerful capability of Eve with $N = 2^{96}$ such that attack success probability increases from 2^{-64} to 2^{-32}, where is observable in our experiments.

4) Evaluation metrics: The evaluation metrics used in our experiments are security loss L_{security} and efficiency loss $L_{\text{efficiency}}$ defined in Section 5.3.2 as well as the bits mismatch rate R_{mismatch}, which is the ratio of the number of bits that do not match between Alice and Bob to the number of bits extracted from channel. All the experimental results are the average value from at least 30 independent experiments.

5.6.2 Evaluation Results

1) Evaluation of existing secret key generation methods: We aim to show the performance (i.e., L_{security} and $L_{\text{efficiency}}$ for 128-bit secret key) of existing methods with the P-value threshold α and R_{privacy} settings in the literature in comparison with the new values for these parameters based on our design guideline. We collect the channel information under 5 meters laboratory environment and using the frequency test.

Figure 5.6a shows the security and efficiency losses of these seven different secret key generation methods before and after new parameter settings based on our guideline. We



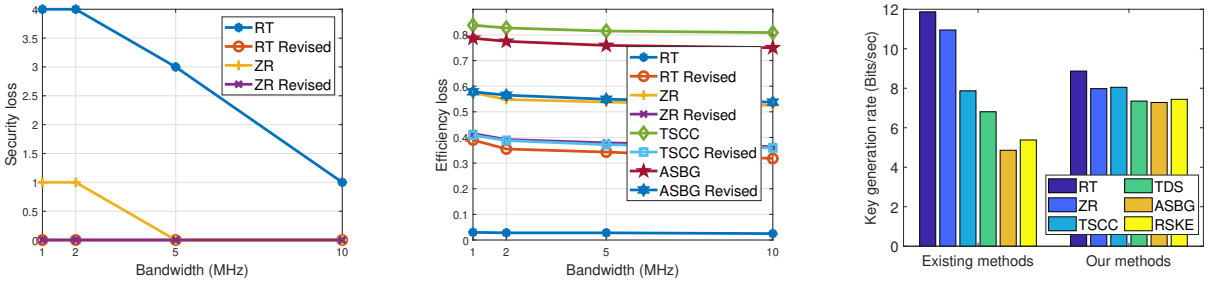
(a) L_{security} under different environments.

(b) $L_{\text{efficiency}}$ under different environments.

(c) L_{security} under multiple level quantizations.

(d) $L_{\text{efficiency}}$ under multiple level quantizations.

Figure 5.7: Experiments under in environments and quantizations.



(a) L_{security} under different bandwidths.

(b) $L_{\text{efficiency}}$ under different bandwidths.

(c) Key generation rate of different methods.

Figure 5.8: Experiments in different bandwidths and methods.

can see that RT, ZR, and TSCC, which set the tests in the wireless domain, have higher L_{security} than others in the cryptographic domain, e.g., L_{security} of RT is 4. Although setting test T in the cryptographic domain can eliminate security loss, it leads to high efficiency loss, e.g, ASBG has $L_{\text{efficiency}} = 0.883$. After the new settings based on our design guideline, i) all of these methods have no L_{security}; ii) $L_{\text{efficiency}}$ significantly decreases. For example, ASBG changes from 0.883 to 0.487 in terms of efficiency loss. Through the revision of parameters based on our guideline, R_{mismatch} of all methods can be reduced, as shown in Figure 5.6b. Figures 5.6c and 5.6d show the P-value threshold α and R_{privacy} are calculated by our design guideline, which indicate that different secret key generation methods need to calculate different α and R_{privacy} to eliminate security loss and achieve high efficiency. On the other hand, Figures 5.6c and 5.6d also illustrate that if we choose a higher P-value threshold, we also need a higher R_{privacy} to eliminate security loss.

2) Evaluation of different experimental environments: The secret key generation method may extract the secret keys with different randomness degrees under varying wireless communication environments. We will show L_{security} and $L_{\text{efficiency}}$ with or without our design guideline in different practical scenarios. Among the 6 secret key generation methods, we select RT and ASBG, since they set the NIST test at different positions. We conduct experiments to collect RSSI, CSI, or phase shift in different indoor environments. For the static USRPs setting, we set the distance of two USRPs to be 5 and 10 meters in the 40m^2 laboratory, 10 and 25 meters in the 350m^2 public lobby, and 25 meters in 450m^2 campus library. For the movement setting, we fix the position of one USRP and randomly move another one in a laboratory. We consider the laboratory environment line-of-sight condition, and lobby and library environments non-line-of-sight condition.

Figure 5.7a shows L_{security} under different experimental environments, and that RT performs the worst for each environment. After revision by our guideline, we eliminate all security loss (i.e., $L_{\text{security}} = 0$). From Figure 5.7b, we increase the $L_{\text{efficiency}}$ from 0.05 to around 0.4 of RT under each communication environment. In ASBG, since they set a small original R_{privacy} value, it incurs high $L_{\text{efficiency}} = \{0.85, 0.74, 0.87, 0.71, 0.68, 0.69\}$ under different environments. After revision by our guideline, we properly set the new R_{privacy} so that ASBG performs more efficiently such that $L_{\text{efficiency}} = \{0.41, 0.49, 0.37, 0.39, 0.39, 0.38\}$.

3) Evaluation of multiple-level quantizations: Figures 5.7c and 5.7d show L_{security} and $L_{\text{efficiency}}$ under m-ary quantization methods ($m \in \{2, 4, 8\}$) with 4 methods: RT, TSCC, ASBG and RSKE. Due to the fact that ASBG and RSKE with original settings have no security loss, we do not show them in Figure 5.7c for clearer illustration. In Figure 5.7c, we can see that a larger value of m incurs a higher security loss (e.g., for RT, $L_{\text{security}} = 7$ when $m = 8$). After revision based on our guideline, we ensure $L_{\text{security}} = 0$ for RT and TSCC. In Figure 5.7d, it is observed that when we ensure L_{security} =0 by setting up randomness tests for different values of m, the efficiency L_{security} approximately remains the same. This

Table 5.2: P-value threshold α and R_{privacy} for different quantization methods.

	RT		ZR		TSCC		TDS		ASBG		RSKE	
	α	R_{privacy}	α	R_{privacy}	α	R_{privacy}	α	R_{privacy}	α	R_{privacy}	α	R_{privacy}
Frequency	0.11	0.50	0.08	0.47	0.17	0.68	0.09	0.48	0.13	0.62	0.16	0.60
Block of Frequency	0.21	0.67	0.18	0.59	0.27	0.79	0.19	0.61	0.25	0.72	0.20	0.63
Run	0.14	0.51	0.09	0.51	0.22	0.70	0.11	0.52	0.14	0.63	0.14	0.57
Longest Run of ones	0.13	0.48	0.18	0.57	0.21	0.66	0.10	0.47	0.17	0.61	0.17	0.66
DFT	0.07	0.61	0.14	0.62	0.14	0.71	0.09	0.54	0.13	0.62	0.11	0.68
Non-overlap	0.08	0.46	0.20	0.55	0.17	0.61	0.17	0.50	0.22	0.66	0.21	0.63
Approximate entropy	0.06	0.41	0.18	0.53	0.14	0.62	0.15	0.51	0.19	0.62	0.15	0.61
First order serial	0.06	0.44	0.11	0.41	0.15	0.61	0.09	0.44	0.11	0.57	0.08	0.59
Second order serial	0.04	0.40	0.09	0.42	0.14	0.60	0.11	0.47	0.08	0.51	0.09	0.61

indicates that simply adopting a multiple-level quantization method does not necessarily mean a faster key generation rate with a security guarantee.

4) Evaluation of different bandwidths: Figures 5.8a and 5.8b show L_{security} and $L_{\text{efficiency}}$ under different bandwidth settings $\{1, 2, 5, 10\}$MHz in the lobby environment. We do not show the L_{security} of ASBG and RSKE, since they have no security loss. When we increase the bandwidth, it is easy to observe that L_{security} decreases under RT and TSCC (e.g., for RT, L_{security} goes from 4 to 1). We can observe that the wireless signal with a larger bandwidth makes the random bit sequence less correlated and therefore more random. From the efficiency perspective, increasing bandwidth can decrease $\mathbb{P}(T \text{ accepts } H_0)$ such that $L_{\text{efficiency}}$ becomes smaller (e.g., for ASBG, $L_{\text{efficiency}}$ decreases from 0.57 to 0.52).

5) Evaluation of key generation rates: For testing the key generation rate of each method, we setup the 6 existing key generation methods for extracting 128-bit key sequences using the parameters in Table 5.1 under 5 meters laboratory environment. Under the same environment, we revise the parameters by our guideline and test the key generation rate. For each experiment, we collect 10,000 bits and calculate the rate (bits/sec). The results are shown in Figure 5.8c. We note that RT and ZR have higher rates with their original setups but they both have security losses. Our setups ensure the maximum key generation rates (e.g., 7.49 bits/sec for ASBG) without security loss.

6) Evaluation of different randomness tests T and R_{privacy}: Table 5.2 indicates that no tests can achieve the optimization goal in (5.7) by using $\alpha = 0.01$. In other words, if we just follow the recommendation of NIST, we may not eliminate the security loss and guarantee

Table 5.3: $L_{\text{efficiency}}$ of different bit sequence length.

	128-bit		256-bit		512-bit	
$L_{\text{efficiency}}$	Original	Revised	Original	Revised	Original	Revised
RT	0.06	0.33	0.03	0.35	0.08	0.31
ZR	0.09	0.39	0.07	0.33	0.05	0.25
TSCC	0.58	0.37	0.57	0.31	0.55	0.24
TDS	0.71	0.57	0.71	0.48	0.73	0.30
ASBG	0.86	0.42	0.88	0.36	0.85	0.29
RSKE	0.73	0.57	0.70	0.44	0.68	0.21

efficiency of secret key generation. On the other hand, if we only focus on security, the efficiency can be very low. Therefore, it is necessary to meet our guideline to consider both efficiency and security aspects. It is noted that Table II shows different values of α and R_{privacy} for different randomness tests. This is due to the fact that each test has its own $\mathbb{P}(T \text{ accept } H_0)$, which is provided in Appendix C.3, in our guideline and we need to solve its own pair of α and R_{privacy} based on the optimization (5.7a).

7) Efficiency of different lengths of secret key generation: Intuitively, if Eve's attack capability does not change, it should be more difficult to crack a longer key sequence. Thus, randomness test T and privacy amplification should be set looser for a longer key sequence. However, the existing methods use the fixed value α and R_{privacy} for generating different lengths of key sequence such that it is detrimental to the secret key generation efficiency (suppose no security loss). We evaluate $L_{\text{efficiency}}$ of 128, 256, and 512-bit secret key generation before and after revision based on our guideline under the laboratory scenario and frequency test. Eve's attack capability is $N = 2^{96}$.

In Table 5.3, we can observe that it is more efficient to generate a longer secret key sequence based on our guideline (e.g., for TDS with revised setups, $L_{\text{efficiency}}$ is 0.57, 0.48 and 0.30 for the 128-bit, 256-bit and 512-bit cases, respectively). However, if we use the parameter setting in the existing studies, $L_{\text{efficiency}}$ does not change obviously, e.g., $L_{\text{efficiency}}$ of RSKE are 0.73, 0.70 and 0.68. Hence, our design guideline also offers an adaptive method to generate keys with different lengths, which is not presented in existing studies.

5.7 Related Work

To provide the confidentiality of data transmission, secret key generation based on the information of wireless channels is promising because of the efficiency and security [194, 123, 188, 98]. In [194], a proximity attack requires a minimal distance from the eavesdropper to maintain perfect secrecy for secret key generation. The randomness test can provide a generic threshold on required distances from an eavesdropper and good key refreshing rates. [123] explores the use of wireless channel characteristics for establishing arbitrary-length secret keys between Bluetooth devices. They verified the output secret bit streams generated by Bluetooth achieve high entropy by the randomness test. [188] tests the randomness of key bits, which quantifies a subcarrier's channel response with different coherent times [98] proposes to defend against threats of eavesdropping and fake data injection in underwater acoustic networks, providing an overview of the advantages of RSSI based key generation and exploring the major challenges from the unique features of acoustic communications.

Key establishment using physical layer characteristics [162, 191, 54], which are a much richer source of secret information but high computational cost overhead. [162] reviews different types of existing methods based on quantization, handling communication errors and the feasibility and security issues related to these methods. The chapter [54] provides an efficient secret key generation method using multipath relative delay from Ultra-wideband (UWB) channels. They study a statistical characterization of UWB channels in a residential scenario and evaluate key mismatch probability. [84] presents the key establishment that uses the distance variation trends caused by the motion paths of two devices to each other.

Recently, some studies have started working on authenticating the transmitter and receiver based on prior coordination or secret sharing. [119] proposes physical-layer authentication schemes through adding low-power signal. [189] solves the authentication in IoT by exploiting the fading of the wireless links between devices to be authenticated and a set of trusted anchor nodes. [4] proposes a retroactive key setup to protect source authentication

and path validation into the realm of practicality. [190, 91] adapts fingerprint embedding to keep message authentication and increased security by obscuring the authentication tag.

5.8 Summary

This chapter studies how to properly test the wireless channel randomness for security and efficiency. In particular, we propose a new design guideline that can choose the P-value threshold, a critical parameter of the randomness test, to ensure the security of the wireless system as well as achieve a high secret bit generation rate with privacy amplification. Since the practical channel information (CSI, RSSI, or phase shifts) is imperfectly independent, we come up with a new cracking key attack called MLTS, which searches the bit sequence by leveraging the Markov-dependent property. By turning a suitable P-value threshold and privacy amplification rate, we formulate an optimization problem to maximize the key generation rate under the constraint of no security loss. Our analysis indicates that the randomness test T should be set in the wireless domain. We conduct different practical environments to validate the analysis of our guideline. By comparing to existing key generation methods, results show that our guideline can improve these methods to be more efficient and secure.

Chapter 6: Conclusion

In this dissertation, we analyze the current learning-based wireless networking performance limitations and security issues. Then, we propose a new framework and client selection policy, especially for the FL learning framework. To this end, we design new defenses and guideline for FL and wireless communication key generation.

For learning-based wireless networking, we first explored the shortcoming of the conventional synchronous single-server network architecture, which must wait for the slowest client and then process the aggregation. However, it cannot be avoidable that some clients locate in complicated environments and perform poor wireless channel conditions. As a result, these clients will prolong the training time. Although some existing studies proposed new FL architecture HFL, they have not presented how to select clients to improve training performance. To address this problem, we propose an online client selection policy called COCS, which leverages the CC-MAB algorithm, and demonstrate our COCS policy can achieve sublinear regret in both convex and non-convex FL settings. Because of the poor convergence performance of HFL, we propose a new multi-server FL architecture and the corresponding FL algorithm, MS-FedAvg, to tackle this issue. Both the theoretical and empirical results show that MS-FedAvg outperforms the existing multi-server FL system.

Although FL can protect users' privacy, model poisoning attacks also successfully degrade the FL performance. Although the Byzantine resilience aggregation rule can protect honest clients, many new attack models have demonstrated that they can easily pass the test. Therefore, we propose a new defense strategy called LoMar, which is based on the kernel function to compare the distribution of local datasets. We present a detailed theoretical analysis and extensive experiments, which indicate that LoMar outperforms the existing FL

defense strategies and sufficiently detects the model poisoning attack. Key generation based on a wireless channel has been widely used in physical layer authentication. We revisit the key generation scheme and find an interesting observation that wireless channel information is not sufficiently random. As such, we propose a new attack that leverages this property to manipulate the secret key. Instead of proposing a new key generation strategy, we design a new guideline for how to use the wireless channel to generate the secret key.

References

[1] Syed Taha Ali, Vijay Sivaraman, and Diethelm Ostry. Eliminating reconciliation cost in secret key generation for body-worn health monitoring devices. *IEEE Trans. Mobile Comput*, 13(12):2763–2776, 2014.

[2] Mohammad Mohammadi Amiri and Deniz Gündüz. Federated learning over wireless fading channels. *IEEE Trans. Wireless Commun.*, 19(5):3546–3557, 2020.

[3] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[4] Katerina Argyraki, Suhas Diggavi, Melissa Duarte, Christina Fragouli, Marios Gatzianas, and Panagiotis Kostopoulos. Creating secrets out of erasures. In *Proceedings of the ACM MobiCom*, pages 429–440, 2013.

[5] Javed A Aslam and Scott E Decatur. On the sample complexity of noise-tolerant learning. *Information Processing Letters*, 57(4):189–195, 1996.

[6] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[7] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[8] Chen Avin, Yuval Emek, Erez Kantor, Zvi Lotker, David Peleg, and Liam Roditty. Sinr diagrams: Towards algorithmically usable sinr models of wireless networks. In *Proceedings of the ACM PODC*, pages 200–209, 2009.

[9] Babak Azimi-Sadjadi, Aggelos Kiayias, Alejandra Mercado, and Bulent Yener. Robust key generation from signal envelopes in wireless networks. In *ACM CCS*, pages 401–410, 2007.

[10] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the SODA*, pages 1497–1514. SIAM, 2014.

[11] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[12] Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, and Jaehoon Amir Safavi. Mitigating poisoning attacks on machine learning models: A data provenance based approach. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 103–110. ACM, 2017.

[13] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[14] Lawrence E Bassham III, Andrew L Rukhin, Juan Soto, James R Nechvatal, Miles E Smid, Elaine B Barker, Stefan D Leigh, Mark Levenson, Mark Vangel, David L Banks, et al. *Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications*. National Institute of Standards & Technology, 2010.

[15] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *ACM SIGKDD explorations newsletter*, 2(2):81–85, 2000.

[16] Charles H Bennett, Gilles Brassard, Claude Crépeau, and Ueli M Maurer. Generalized privacy amplification. *IEEE Trans. Information Theory*, 41(6):1915–1923, 1995.

[17] Arjun Nitin Bhagoji, Supriyo Chakraborty, Seraphin Calo, and Prateek Mittal. Model poisoning attacks in federated learning. In *In Workshop on Security in Machine Learning (SecML), collocated with the 32nd Conference on Neural Information Processing Systems (NeurIPS'18)*, 2018.

[18] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *ICML*, pages 634–643, 2019.

[19] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on ICML*, pages 1467–1474, 2012.

[20] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, pages 119–129, 2017.

[21] Gilles Brassard and Louis Salvail. Secret-key reconciliation by public discussion. In *Workshop on Theory and Application of Cryptographic Techniques*, pages 410–423, 1993.

[22] Leo Breiman, William Meisel, and Edward Purcell. Variable kernel estimates of multivariate densities. *Technometrics*, 19(2):135–144, 1977.

[23] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018.

[24] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.

[25] C. Chen and M. A. Jensen. Secret key establishment using temporally and spatially correlated wireless channel coefficients. *IEEE Trans. Mobile Comput*, 10(2):205–215, Feb 2011.

[26] Chan Chen and Michael A Jensen. Secret key establishment using temporally and spatially correlated wireless channel coefficients. *IEEE Trans. Mobile Comput*, 10(2):205–215, 2011.

[27] Lin Chen, Andreas Krause, and Amin Karbasi. Interactive submodular bandit. In *NIPS*, pages 141–152, 2017.

[28] Lixing Chen, Zhuo Lu, Pan Zhou, and Jie Xu. Learning optimal sniffer channel assignment for small cell cognitive radio networks. In *IEEE INFOCOM*, pages 656–665, 2020.

[29] Lixing Chen and Jie Xu. Budget-constrained edge service provisioning with demand estimation via bandit learning. *IEEE J. Sel. Areas Commun.*, 37(10):2364–2376, 2019.

[30] Hoesang Choi and Hichan Moon. Throughput of cdm-based random access with sinr capture. *IEEE Trans. Veh. Technol.*, 69(12):15046–15056, 2020.

[31] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *IJCNN*, pages 2921–2926. IEEE, 2017.

[32] IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[33] Guangming Cui, Qiang He, Xiaoyu Xia, Feifei Chen, Hai Jin, and Yun Yang. Robustness-oriented k edge server placement. In *IEEE CCGRID*, pages 81–90, 2020.

[34] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM*, volume 34, pages 15–26, 2004.

[35] Elizabeth R DeLong, David M DeLong, and Daniel L Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845, 1988.

[36] Yucheng Ding, Chaoyue Niu, Yikai Yan, Zhenzhe Zheng, Fan Wu, Guihai Chen, Shaojie Tang, and Rongfei Jia. Distributed optimization over block-cyclic data. *arXiv preprint arXiv:2002.07454*, 2020.

[37] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[38] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Trans. Parallel Distrib. Syst.*, 32(1):59–71, 2020.

[39] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Accurate federated learning via decomposed similarity-based clustering. *arXiv preprint arXiv:2010.06870*, 2020.

[40] Rui Duan, Zhe Qu, Shangqing Zhao, Leah Ding, Yao Liu, and Zhuo Lu. Perception-aware attack: Creating adversarial music via reverse-engineering human perception. *arXiv preprint arXiv:2207.13192*, 2022.

[41] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Louis Alexandre Rouault. The hidden vulnerability of distributed learning in byzantium. In *ICML*, number CONF, 2018.

[42] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1605–1622, 2020.

[43] Yann Fraboni, Richard Vidal, and Marco Lorenzi. Free-rider attacks on model aggregation in federated learning. In *AISTATS*, pages 1846–1854. PMLR, 2021.

[44] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv e-prints*, pages arXiv–1808, 2018.

[45] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *NeurIPS*, 33, 2020.

[46] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

[47] Pengfei Guo, Puyang Wang, Jinyuan Zhou, Shanshan Jiang, and Vishal M Patel. Multi-institutional collaborations for improving deep learning-based magnetic resonance image reconstruction using federated learning. In *Proceedings of the IEEE/CVF CVPR*, pages 2423–2432, 2021.

[48] Shuo Guo, Heng Zhang, Ziguo Zhong, Jiming Chen, Qing Cao, and Tian He. Detecting faulty nodes with data errors for wireless sensor networks. *ACM TOSN*, 10(3):1–27, 2014.

[49] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In *AISTATS*, pages 2350–2358. PMLR, 2021.

[50] Dong-Jun Han, Minseok Choi, Jungwuk Park, and Jaekyun Moon. Fedmes: Speeding up federated learning with multiple edge servers. *IEEE J. Sel. Areas Commun.*, 39(12):3870–3885, 2021.

[51] Val Henson. An analysis of compare-by-hash. In *HotOS*, pages 13–18, 2003.

[52] Tao Hou, Zhe Qu, Tao Wang, Zhuo Lu, and Yao Liu. Proto: Proactive topology obfuscation against adversarial network topology inference. In *IEEE INFOCOM*, pages 1598–1607, 2020.

[53] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[54] Jingjing Huang and Ting Jiang. Dynamic secret key generation exploiting ultra-wideband wireless channel characteristics. In *IEEE WCNC*, pages 1701–1706, 2015.

[55] Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Zomaya. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. *IEEE Trans. Parallel Distrib. Syst.*, 2020.

[56] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *NeurIPS*, 33, 2020.

[57] Xiaozhou Huang and Hsiao-Chun Wu. Robust and efficient intercarrier interference mitigation for ofdm systems in time-varying fading channels. *IEEE Trans. Veh. Technol.*, 56(5):2517–2528, 2007.

[58] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE S&P*, pages 19–35, 2018.

[59] Suman Jana, Sriram Nandha Premnath, Mike Clark, Sneha K Kasera, Neal Patwari, and Srikanth V Krishnamurthy. On the effectiveness of secret key extraction from wireless signal strength in real environments. In *ACM MOBICOM*, pages 321–332, 2009.

[60] Galin L Jones et al. On the markov chain central limit theorem. *Probability surveys*, 1(299-320):5–1, 2004.

[61] 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. *arXiv preprint arXiv:1912.04977*, 2019.

[62] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *ICML*, pages 5132–5143. PMLR, 2020.

[63] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network security: private communication in a public world*. Prentice Hall Press, 2002.

[64] Hwanjin Kim and Junil Choi. Channel estimation for one-bit massive mimo systems exploiting spatio-temporal correlations. In *IEEE GLOBECOM*, pages 1–6, 2018.

[65] Jakub Konecny, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2017.

[66] Jakub Konevcny, H Brendan McMahan, Felix X Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[67] Lingjing Kong, Tao Lin, Anastasia Koloskova, Martin Jaggi, and Sebastian Stich. Consensus control for decentralized deep learning. In *Proceedings of the 38th ICML*, volume 139, pages 5686–5696. PMLR, 2021.

[68] Xiangjie Kong, Kailai Wang, Mingliang Hou, Xinyu Hao, Guojiang Shen, Xin Chen, and Feng Xia. A federated learning-based license plate recognition scheme for 5g-enabled internet of vehicles. *IEEE Trans. Ind. Informat.*, 17(12):8523–8530, 2021.

[69] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[70] Yang Kuang, Tingwen Ruan, Zheng Jun Chew, and Meiling Zhu. Energy harvesting during human walking to power a wireless sensor node. *Sensors and Actuators A: Physical*, 254:69–77, 2017.

[71] John B Lacy, Donald P Mitchell, and William M Schell. Cryptolib: Cryptography in software. In *USENIX Security Symposium*, 1993.

[72] Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang. Optimal edge user allocation in edge computing with variable sized vector bin packing. In *ICSOC*, pages 230–245. Springer, 2018.

[73] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[74] Jin-woo Lee, Jaehoon Oh, Sungsu Lim, Se-Young Yun, and Jae-Gil Lee. Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture. *arXiv preprint arXiv:2012.03214*, 2020.

[75] Jon Lee. *A first course in combinatorial optimization*. Number 36. Cambridge University Press, 2004.

[76] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[77] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[78] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[79] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[80] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *ICLR*, 2019.

[81] Xingyu Li, Zhe Qu, Bo Tang, and Zhuo Lu. Fedlga: Towards system-heterogeneity of federated learning via local gradient approximation. *arXiv preprint arXiv:2112.11989*, 2021.

[82] Xingyu Li, Zhe Qu, Bo Tang, and Zhuo Lu. Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients. *arXiv preprint arXiv:2102.06329*, 2021.

[83] Xingyu Li, Zhe Qu, Shangqing Zhao, Bo Tang, Zhuo Lu, and Yao Liu. Lomar: A local defense against poisoning attack on federated learning. *IEEE Trans. Dependable Secure Comput.*, 2021.

[84] Zi Li, Qingqi Pei, Ian Markwood, Yao Liu, and Haojin Zhu. Secret key establishment via rss trajectory matching between wearable devices. *IEEE Trans. Inf. Forensics Security*, 13(3):802–817, 2017.

[85] Yingbin Liang, H Vincent Poor, and Shlomo Shamai. Secure communication over fading channels. *IEEE Trans. Inf. Theory*, 54(6):2470–2492, 2008.

[86] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Commun. Surveys Tuts.*, 22(3):2031–2063, 2020.

[87] Bo Lindqvist. A note on bernoulli trials with dependence. *Scandinavian Journal of Statistics*, pages 205–208, 1978.

[88] Chen-Feng Liu, Mehdi Bennis, Merouane Debbah, and H Vincent Poor. Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. *IEEE Trans. Commun.*, 67(6):4132–4150, 2019.

[89] Hongbo Liu, Yang Wang, Jie Yang, and Yingying Chen. Fast and practical secret key extraction by exploiting channel response. In *IEEE INFOCOM*, pages 3048–3056, 2013.

[90] Hongbo Liu, Jie Yang, Yan Wang, and Yingying Chen. Collaborative secret key extraction leveraging received signal strength in mobile wireless networks. In *IEEE INFOCOM*, pages 927–935, 2012.

[91] Jiazi Liu and Xianbin Wang. Physical layer authentication enhancement using two-dimensional channel quantization. *IEEE Trans. Wireless Commun.*, 15(6):4171–4182, 2016.

[92] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. Client-edge-cloud hierarchical federated learning. In *IEEE ICC*, pages 1–6, 2020.

[93] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B Letaief. Hierarchical quantized federated learning: Convergence analysis and system design. *arXiv preprint arXiv:2103.14272*, 2021.

[94] Wanchun Liu, Xiangyun Zhou, Salman Durrani, and Petar Popovski. Secure communication with a wireless-powered friendly jammer. *IEEE Trans. Wireless Commun.*, 15(1):401–415, 2015.

[95] Xian Liu, Dawei Lu, Aiqian Zhang, Qian Liu, and Guibin Jiang. Data-driven machine learning in environmental pollution: Gains and problems. *Environmental Science & Technology*, 2022.

[96] Don O Loftsgaarden, Charles P Quesenberry, et al. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051, 1965.

[97] Siqi Luo, Xu Chen, Qiong Wu, Zhi Zhou, and Shuai Yu. Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Trans. Wireless Commun.*, 19(10):6535–6548, 2020.

[98] Yu Luo, Lina Pu, Zheng Peng, and Zhijie Shi. Rss-based secret key generation in underwater acoustic networks: advantages, challenges, and performance improvements. *IEEE Communications Magazine*, 54(2):32–38, 2016.

[99] Zhengping Luo, Zhe Qu, Tung Nguyen, Hui Zeng, and Zhuo Lu. Security of hpc systems: From a log-analyzing perspective. *EAI Endorsed Transactions on Security and Safety*, 6(21):e5, 2019.

[100] Jeaneth Machicao and Odemir M Bruno. Improving the pseudo-randomness properties of chaotic maps using deep-zoom. *Chaos: an interdisciplinary journal of nonlinear science*, 27(5):053116, 2017.

[101] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surveys Tuts.*, 19(4):2322–2358, 2017.

[102] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[103] Suhas Mathur, Wade Trappe, Narayan Mandayam, Chunxuan Ye, and Alex Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *ACM MOBICOM*, pages 128–139, 2008.

[104] Ueli Maurer and Stefan Wolf. Secret-key agreement over unauthenticated public channels - Part III: Privacy amplification. *IEEE Trans. Information Theory*, 49(4):839–851, 2003.

[105] Colin McDiarmid. Concentration. In *Probabilistic methods for algorithmic discrete mathematics*, pages 195–248. Springer, 1998.

[106] Matthew R McKay and Iain B Collings. General capacity bounds for spatially correlated rician mimo channels. *IEEE Trans. Inf. Theory*, 51(9):3121–3145, 2005.

[107] 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*, pages 1273–1282. PMLR, 2017.

[108] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE S&P*, pages 691–706, 2019.

[109] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE S&P*, pages 739–753, 2019.

[110] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NIPS*, pages 1196–1204, 2013.

[111] Hung T Nguyen, Vikash Sehwag, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor. Fast-convergent federated learning. *IEEE J. Sel. Areas Commun.*, 39(1):201–218, 2020.

[112] Ti Ti Nguyen, Vu Nguyen Ha, Long Bao Le, and Robert Schober. Joint data compression and computation offloading in hierarchical fog-cloud systems. *IEEE Trans. Wireless Commun.*, 19(1):293–309, 2019.

[113] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *IEEE ICC*, pages 1–7, 2019.

[114] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of the ACM MobiSys*, pages 54–66, 2021.

[115] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the ACM ASIACCS*, pages 506–519, 2017.

[116] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. Sok: Security and privacy in machine learning. In *IEEE EuroS&P*, pages 399–414, 2018.

[117] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[118] Neal Patwari, Jessica Croft, Suman Jana, and Sneha K Kasera. High-rate uncorrelated bit extraction for shared secret key generation from channel measurements. *IEEE Trans. Mobile Comput*, 9(1):17–30, 2010.

[119] L Yu Paul, John S Baras, and Brian M Sadler. Physical-layer authentication. *IEEE Trans. Inf. Forensics Security*, 3(1):38–51, 2008.

[120] Craig S Petrie and J Alvin Connelly. A noise-based ic random number generator for applications in cryptography. *IEEE Trans. Circuits Syst. I. Fundam. Theory Appl.*, 47(5):615–621, 2000.

[121] Satya Ponnaluri, Babak Azimi-Sadjadi, Yik-Kiong Hue, Tugba Erpek, Arash Komaee, and Wade Trappe. A practical wireless reciprocity-aware key establishment protocol. In *IEEE MILCOM*, pages 1107–1113, 2016.

[122] S. N. Premnath, P. L. Gowda, S. K. Kasera, N. Patwari, and R. Ricci. Secret key extraction using bluetooth wireless signal strength measurements. In *IEEE SECON*, pages 293–301, 2014.

[123] Sriram Nandha Premnath, Jessica Croft, Neal Patwari, and Sneha Kumar Kasera. Efficient high-rate secret key extraction in wireless sensor networks using collaboration. *ACM TOSN*, 11(1):2, 2014.

[124] Xiao Qin, Lei Cao, Elke A Rundensteiner, and Samuel Madden. Scalable kernel density estimation-based local outlier detection over large data streams. In *EDBT*, pages 421–432, 2019.

[125] Zhaonan Qu, Kaixiang Lin, Jayant Kalagnanam, Zhaojian Li, Jiayu Zhou, and Zhengyuan Zhou. Federated learning's blessing: Fedavg has linear speedup. *arXiv preprint arXiv:2007.05690*, 2020.

[126] Zhe Qu, Rui Duan, Lixing Chen, Jie Xu, Zhuo Lu, and Yao Liu. Context-aware online client selection for hierarchical federated learning. *arXiv preprint arXiv:2112.00925*, 2021.

[127] Zhe Qu, Rui Duan, Lixing Chen, Jie Xu, Zhuo Lu, and Yao Liu. Context-aware online client selection for hierarchical federated learning. *arXiv preprint arXiv:2112.00925*, 2021.

[128] Zhe Qu, Rui Duan, Lixing Chen, Jie Xu, Zhuo Lu, and Yao Liu. Context-aware online client selection for hierarchical federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 2022.

[129] Zhe Qu, Xingyu Li, Rui Duan, Yao Liu, Bo Tang, and Zhuo Lu. Generalized federated learning via sharpness aware minimization. *arXiv preprint arXiv:2206.02618*, 2022.

[130] Zhe Qu, Xingyu Li, Jie Xu, Bo Tang, Zhuo Lu, and Yao Liu. On the convergence of multi-server federated learning with overlapping area. *IEEE Transactions on Mobile Computing*, 2022.

[131] Zhe Qu, Shangqing Zhao, Jie Xu, Zhuo Lu, and Yao Liu. How to test the randomness from the wireless channel for security? *IEEE Transactions on Information Forensics and Security*, 16:3753–3766, 2021.

[132] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konevcny, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *ICLR*, 2021.

[133] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *AISTATS*, pages 2021–2031. PMLR, 2020.

[134] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[135] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE CVPR*, pages 4510–4520, 2018.

[136] Timothy M Schmidl and Donald C Cox. Robust frequency and timing synchronization for ofdm. *IEEE Trans. Commun.*, 45(12):1613–1621, 1997.

[137] Erich Schubert, Remigius Wojdanowski, Arthur Zimek, and Hans-Peter Kriegel. On evaluation of outlier rankings and outlier scores. In *Proceedings of SDM*, pages 1047–1058. SIAM, 2012.

[138] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*, pages 6103–6113, 2018.

[139] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the ACSAC*, pages 508–519, 2016.

[140] Wenqi Shi, Sheng Zhou, and Zhisheng Niu. Device scheduling with fast convergence for wireless federated learning. In *IEEE ICC*, pages 1–6, 2020.

[141] Yuanming Shi, Kai Yang, Tao Jiang, Jun Zhang, and Khaled B Letaief. Communication-efficient edge ai: Algorithms and systems. *IEEE Commun. Surveys Tuts.*, 22(4):2167–2191, 2020.

[142] Nir Shlezinger, Mingzhe Chen, Yonina C Eldar, H Vincent Poor, and Shuguang Cui. Uveqfed: Universal vector quantization for federated learning. *IEEE Trans. Signal Process.*, 69:500–514, 2020.

[143] David Slepian and Jack Wolf. Noiseless coding of correlated information sources. *IEEE Trans. Inf. Theory*, 19(4):471–480, 1973.

[144] Guillaume Stempfel, Liva Ralaivola, and François Denis. Learning from Noisy Data using Hyperplane Sampling and Sample Averages, May 2007. working paper or preprint.

[145] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full sha-1. In *Crypto*, pages 570–596. Springer, 2017.

[146] Sebastian U Stich. Local sgd converges fast and communicates little. In *ICLR*, 2018.

[147] Haijian Sun, Xiang Ma, and Rose Qingyang Hu. Adaptive federated learning with gradient compression in uplink noma. *IEEE Trans. Veh. Technol.*, 69(12):16325–16329, 2020.

[148] Bo Tang and Haibo He. Kerneladasyn: Kernel based adaptive synthetic data generation for imbalanced learning. In *IEEE CEC*, pages 664–671, 2015.

[149] Bo Tang and Haibo He. A local density-based approach for outlier detection. *Neurocomputing*, 241:171–180, 2017.

[150] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *PAKDD*, pages 535–548. Springer, 2002.

[151] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *ESORICS*, pages 480–501. Springer, 2020.

[152] Nguyen H Tran, Wei Bao, Albert Zomaya, Minh NH Nguyen, and Choong Seon Hong. Federated learning over wireless networks: Optimization model design and analysis. In *IEEE INFOCOM*, pages 1387–1395, 2019.

[153] Tuyen X Tran and Dario Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Trans. Veh. Technol.*, 68(1):856–868, 2018.

[154] Gill R Tsouri and David M Wagner. Threshold constraints on symmetric key extraction from rician fading estimates. *IEEE Trans. Mobile Comput*, 12(12):2496–2506, 2013.

[155] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A McKay, Mary L Baish, and Mike Boyle. Recommendation for the entropy sources used for random bit generation. *NIST Special Publication*, 800:90B, 2018.

[156] Jon W Wallace and Rajesh K Sharma. Automatic secret keys from reciprocal MIMO wireless channels: Measurement and analysis. *IEEE Trans. Inf. Forensics Security*, 5(3):381–392, 2010.

[157] Ge Wang, Haofan Cai, Chen Qian, Jinsong Han, Xin Li, Han Ding, and Jizhong Zhao. Towards replay-resilient rfid authentication. In *Proceedings of the ACM MobiCom*, pages 385–399, 2018.

[158] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM*, pages 1698–1707, 2020.

[159] Jiayi Wang, Shiqiang Wang, Rong-Rong Chen, and Mingyue Ji. Local averaging helps: Hierarchical federated learning and convergence analysis. *arXiv preprint arXiv:2010.12998*, 2020.

[160] Qian Wang, Hai Su, Kui Ren, and Kwangjo Kim. Fast and scalable secret key generation exploiting channel phase randomness in wireless networks. In *IEEE INFOCOM*, pages 1422–1430, 2011.

[161] Qian Wang, Kaihe Xu, and Kui Ren. Cooperative secret key generation from phase estimation in narrowband fading channels. *IEEE J. Selected Areas in Communications*, 30(9):1666–1674, 2012.

[162] Tao Wang, Yao Liu, and Athanasios V Vasilakos. Survey on channel reciprocity based key establishment techniques for wireless systems. *Wireless Networks*, 21(6):1835–1846, 2015.

[163] Tian Wang, Lei Qiu, Arun Kumar Sangaiah, Anfeng Liu, Md Zakirul Alam Bhuiyan, and Ying Ma. Edge-computing-based trustworthy data collection model in the internet of things. *IEEE Internet Things J.*, 7(5):4218–4227, 2020.

[164] YH Wang and Zejiang Yang. On a markov multinomial distribution. *Mathematical Scientist*, 20(1):40–49, 1995.

[165] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Cailian Chen, Shi Jin, Zhu Han, and H Vincent Poor. Low-latency federated learning over wireless channels with differential privacy. *arXiv preprint arXiv:2106.13039*, 2021.

[166] Yunchuan Wei, Kai Zeng, and Prasant Mohapatra. Adaptive wireless channel probing for shared key generation based on PID controller. *IEEE Trans. Mobile Comput*, 12(9):1842–1852, 2013.

[167] Jinze Wu, Qi Liu, Zhenya Huang, Yuting Ning, Hao Wang, Enhong Chen, Jinfeng Yi, and Bowen Zhou. Hierarchical personalized federated learning for user modeling. In *Proceedings of the Web Conference 2021*, pages 957–968, 2021.

[168] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Trans. Comput.*, 70(5):655–668, 2020.

[169] Wei Xi, Chen Qian, Jinsong Han, Kun Zhao, Sheng Zhong, Xiang-Yang Li, and Jizhong Zhao. Instant and robust authentication and key agreement among mobile devices. In *ACM CCS*, pages 616–627, 2016.

[170] Wenchao Xia, Tony QS Quek, Kun Guo, Wanli Wen, Howard H Yang, and Hongbo Zhu. Multi-armed bandit-based client scheduling for federated learning. *IEEE Trans. Wireless Commun.*, 19(11):7108–7123, 2020.

[171] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI*, pages 870–875, 2012.

[172] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.

[173] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *ICML*, pages 6893–6901, 2019.

[174] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning. *arXiv preprint arXiv:2005.01026*, 2020.

[175] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19, 2021.

[176] Jie Xu and Heqiang Wang. Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. *IEEE Trans. Wireless Commun*, 20(2):1188–1200, 2020.

[177] Jing Xu, Sen Wang, Liwei Wang, and Andrew Chi-Chih Yao. Fedcm: Federated learning with client-level momentum. *arXiv preprint arXiv:2106.10874*, 2021.

[178] Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data. In *Proceedings of the Web Conference 2021*, pages 935–946, 2021.

[179] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *NDSS*. Internet Society, 2017.

[180] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *NDSS*, 2017.

[181] Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-{iid} federated learning. In *ICLR*, 2021.

[182] Kai Yang, Tao Jiang, Yuanming Shi, and Zhi Ding. Federated learning via over-the-air computation. *IEEE Trans. Wireless Commun.*, 19(3):2022–2035, 2020.

[183] Shengwen Yang, Bing Ren, Xuhui Zhou, and Liping Liu. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019.

[184] Shimpei Yasukawa, Hisato Iwai, and Hideichi Sasaoka. A secret key agreement scheme with multi-level quantization and parity check using fluctuation of radio channel property. In *IEEE ISIT*, pages 732–736, 2008.

[185] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, pages 5650–5659, 2018.

[186] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI*, volume 33, pages 5693–5700, 2019.

[187] Kai Zeng, Daniel Wu, An Chan, and Prasant Mohapatra. Exploiting multiple-antenna diversity for shared secret key generation in wireless networks. In *IEEE INFOCOM*, pages 1–9, 2010.

[188] Junqing Zhang, Alan Marshall, Roger Woods, and Trung Q Duong. Efficient key generation by exploiting randomness from channel responses of individual ofdm subcarriers. *IEEE Trans. Commun.*, 64(6):2578–2588, 2016.

[189] Mengyang Zhao, Aadarsh Jha, Quan Liu, Bryan A Millis, Anita Mahadevan-Jansen, Le Lu, Bennett A Landman, Matthew J Tyskac, and Yuankai Huo. Faster mean-shift: Gpu-accelerated embedding-clustering for cell segmentation and tracking. *arXiv preprint arXiv:2007.14283*, 2020.

[190] Shangqing Zhao, Zhuo Lu, Zhengping Luo, and Yao Liu. Orthogonality-sabotaging attacks against ofdma-based wireless networks. In *IEEE INFOCOM*, pages 1603–1611, 2019.

[191] Shangqing Zhao, Zhuo Lu, and Cliff Wang. When seeing isn't believing: On feasibility and detectability of scapegoating in network tomography. In *IEEE ICDCS*, pages 172–182, 2017.

[192] Shangqing Zhao, Zhuo Lu, and Cliff Wang. How can randomized routing protocols hide flow information in wireless networks? *IEEE Trans. Wireless Commun.*, 19, 2020.

[193] Shangqing Zhao, Zhengping Luo, Zhuo Lu, Xiang Lu, and Yao Liu. Stateful inter-packet signal processing for wireless networking. In *ACM MobiCom*, 2017.

[194] Shangqing Zhao, Zhe Qu, Zhuo Lu, and Tao Wang. Spectrum tomography attacks: Inferring spectrum allocation mechanisms in multicarrier systems. In *IEEE DySPAN*, pages 1–2, 2019.

[195] Shangqing Zhao, Zhe Qu, Zhengping Luo, Zhuo Lu, and Yao Liu. Comb decoding towards collision-free wifi. In *USENIX NSDI*, 2020.

[196] Yuchen Zhao, Hanyang Liu, Honglin Li, Payam Barnaghi, and Hamed Haddadi. Semi-supervised federated learning for activity recognition. *arXiv preprint arXiv:2011.00851*, 2020.

[197] Sihui Zheng, Cong Shen, and Xiang Chen. Design and analysis of uplink and downlink communications for federated learning. *IEEE J. Sel. Areas Commun.*, 39(7):2150–2167, 2020.

[198] Xiaojun Zhu, Fengyuan Xu, Edmund Novak, Chiu C Tan, Qun Li, and Guihai Chen. Extracting secret key from wireless link dynamics in vehicular environments. In *IEEE INFOCOM*, pages 2283–2291, 2013.

[199] Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. Empirical studies of institutional federated learning for natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 625–634, 2020.

Appendix A: Proofs of Chapter 2

In this appendix, we present all lemma and theorems in Chapter 2 in detail.

A.1 Proof of Lemma 1

In order to simplify the notations, we write $\mathbf{w} = \mathbf{w}_m$ in all the proofs.

Proof. For any client $i \in \mathcal{N}_\theta$ and $e \in [E]$, we have:

$$\begin{aligned}
&\mathbb{E}\|\mathbf{w}_\theta^{t,e} - \mathbf{w}^t\|^2 = \mathbb{E}||\mathbf{w}_i^{t,e-1} - \mathbf{w}^t - \eta_l \mathbf{g}_i^{t,e-1}\|^2 \\
&\leq \mathbb{E}\|\mathbf{w}_i^{t,e-1} - \mathbf{w}^t - \eta_l(\mathbf{g}_i^{t,e-1} - \nabla F_i(\mathbf{w}_i^{t,e-1}) + \nabla F_i(\mathbf{w}_i^{t,e-1}) \\
&- \nabla F_i(\mathbf{w}^t) + \nabla F_i(\mathbf{w}^t) - \nabla f(\mathbf{w}^t) + \nabla f(\mathbf{w}^t))\|^2 \\
&\leq \left(1 + \frac{1}{2E-1} + 6E\eta_l^2 L^2\right)\mathbb{E}\|\mathbf{w}_i^{t,e-1} - \mathbf{w}^t\|^2 + \eta_l^2\sigma^2 + \frac{6E\eta_l^2 N_\theta}{N}\alpha_\theta^2 + 6E\eta_l^2\|\nabla f(\mathbf{w}^t)\|^2 \\
&\leq \left(1 - \frac{1}{E-1}\right)\mathbb{E}\|\mathbf{w}_i^{t,e-1} - \mathbf{w}^t\|^2 + \eta_l^2\sigma^2 + \frac{6E\eta_l^2 N_\theta}{N}\alpha_\theta^2 + 6E\eta_l^2\|\nabla f(\mathbf{w}^t)\|^2.
\end{aligned}$$

Re-rolling the above inequality of the local epoch e, we obtain the following:

$$\begin{aligned}
\frac{1}{N_\theta}\sum_{i\in\mathcal{N}_\theta}\mathbb{E}\|\mathbf{w}_i^{t,e} - \mathbf{w}^t\|^2 &\leq \sum_{\gamma=0}^{e-1}\left(1 + \frac{1}{E-1}\right)^\gamma\left(\eta_l^2\beta_\theta^2 + 6E\eta_l^2\|\nabla f(\mathbf{w}^t)\|^2\right) \\
&\leq 5E\eta_l^2\beta_\theta^2 + 30E^2\eta_l^2\|\nabla f(\mathbf{w}^t)\|^2,
\end{aligned}$$

where $\beta_\theta^2 = \sigma^2 + \frac{6EN_\theta}{N}\alpha_\theta^2$ in short. □

A.2 Proof of Theorem 1

Proof. Based on the smoothness in Assumption 1, taking expectation of $f(\mathbf{w}^{t+1})$ over the randomness in communication round t, we have:

$$\begin{aligned}\mathbb{E}[f(\mathbf{w}^{t+1})] &\leq f(\mathbf{w}^t) - E\eta_g\eta_l||\nabla f(\mathbf{w}^t)||^2 \\ &+ \underbrace{\langle \eta_g \nabla f(\mathbf{w}^t), \mathbb{E}[(\mathbf{w}^{t+1} - \mathbf{w}^t) + E\eta_l \nabla f(\mathbf{w}^t)]\rangle}_{A_1} + \frac{L}{2}\eta_g^2 \underbrace{\mathbb{E}||\mathbf{w}^{t+1} - \mathbf{w}^t||^2}_{A_2}. \end{aligned} \tag{A.1}$$

Now, we bound the term A_1 as follows:

$$\begin{aligned} A_1 &= \langle \nabla f(\mathbf{w}^t), \mathbb{E}[(\mathbf{w}^{t+1} - \mathbf{w}^t) + E\eta_l \nabla f(\mathbf{w}^t)]\rangle \\ &= \left\langle \nabla f(\mathbf{w}^t), \mathbb{E} \sum_{\theta \subseteq 2^{\mathcal{M}}} \left[-\sum_{i\in\theta} \frac{1}{N_\theta} \sum_{e=0}^{E-1} \eta_l \nabla F_i(\mathbf{w}_{t,e}^i) + E\eta_l \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{1}{N_\theta} \sum_{i\in\theta} \nabla F_i(\mathbf{w}^t) \right] \right\rangle \\ &= \frac{E\eta_l}{2}\|\nabla f(\mathbf{w}^t)\|^2 + \frac{\eta_l}{2EN_\theta^2}\mathbb{E}\Bigg[\Bigg\| \sum_{\theta\subseteq 2^{\mathcal{M}}} \sum_{i\in\theta} \sum_{e=0}^{E-1} (\nabla F_i(\mathbf{w}_i^{t,e}) \\ &- \nabla F_i(\mathbf{w}_i^t)) \Bigg\|^2 - \frac{\eta_l}{2EN_\theta^2}\mathbb{E}\Bigg\| \sum_{\theta\subseteq 2^{\mathcal{M}}} \sum_{i\in\theta} \sum_{e=0}^{E-1} \nabla F_i(\mathbf{w}_i^{t,e}) \Bigg\|^2 \\ &\leq E\eta_l \left(\frac{1}{2} + 15E^2L^2\eta_l^2 \right) \|\nabla f(\mathbf{w}^t)\|^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{5N_\theta E^2 L^2 \eta_l^3}{2N} \beta_\theta^2 - \frac{\eta_l}{2EN_\theta^2}\mathbb{E}\Bigg\| \sum_{\theta\subseteq 2^{\mathcal{M}}} \sum_{i\in\theta} \sum_{e=0}^{E-1} \nabla F_i(\mathbf{w}_i^{t,e}) \Bigg\|^2, \end{aligned}$$

where the last inequality is from Lemma 1. The term A_2 can be bounded as follows:

$$A_2 \leq \frac{\eta_l^2}{N_\theta^2}\mathbb{E}\Bigg\| \sum_{\theta\subseteq 2^{\mathcal{M}}} \sum_{i\in\theta} \sum_{e=0}^{E-1} \mathbf{g}_i^{t,e} \Bigg\|^2 \leq \frac{E\eta_l^2}{N_\theta}\sigma^2 + \frac{\eta_l^2}{N_\theta^2}\mathbb{E}\Bigg\| \sum_{\theta\subseteq 2^{\mathcal{M}}} \sum_{i\in\theta} \sum_{e=0}^{E-1} \nabla F_i(\mathbf{w}_i^{t,e}) \Bigg\|^2,$$

where the last inequality is from the Assumption 2 and the fact $\mathbb{E}(\|\mathbf{x}_1 + \cdots + \mathbf{x}_n\|^2) = \mathbb{E}(\|\mathbf{x}_1\|^2 + \cdots + \|\mathbf{x}_n\|^2)$ if each $\mathbf{x}$ is independent with zero mean and $\mathbb{E}[\mathbf{g}_i^{t,e}] = \nabla F_i(\mathbf{w}_i^{t,e})$.

Plugging terms A_1 and A_2 into (A.1) as follows:

$$\begin{aligned}
\mathbb{E}[f(\mathbf{w}^{t+1})] &\leq f(\mathbf{w}^t) - E\eta_l\eta_g\left(\frac{1}{2} - 15E^2L^2\eta_l^2\right)\|\nabla f(\mathbf{w}^t)\|^2 \\
&+ \frac{LE\eta_l^2\eta_g^2}{2N_\theta}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{5N_\theta E^2L^2\eta_g\eta_l^3}{2N}\beta_\theta^2 + \left(\frac{L\eta_g^2\eta_l^2}{2N_\theta^2} - \frac{\eta_g\eta_l}{2EN^2}\right)\mathbb{E}\left\|\sum_{\theta\subseteq 2^{\mathcal{M}}}\sum_{i\in\theta}\sum_{e=0}^{E-1}\nabla F_i(\mathbf{w}_i^{t,e})\right\|^2 \\
&\overset{(1)}{\leq} f(\mathbf{w}^t) - E\eta_l\eta_g\left(\frac{1}{2} - 15E^2L^2\eta_l^2\right)\|\nabla f(\mathbf{w}^t)\|^2 + \frac{LE\eta_l^2\eta_g^2}{2N_\theta}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{5N_\theta E^2L^2\eta_g\eta_l^3}{2N}\beta_\theta^2 \\
&\overset{(2)}{\leq} f(\mathbf{w}^t) - cE\eta_g\eta_l\|\nabla f(\mathbf{w}^t)\|^2 + \frac{LE\eta_l^2\eta_g^2}{2N}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{5N_\theta E^2L^2\eta_g\eta_l^3}{2N}\beta_\theta^2,
\end{aligned}$$

where (1) is based on $\eta_g\eta_l \leq \frac{1}{LE}$ and (2) is based on $\eta_l \leq \frac{1}{\sqrt{30}LE}$. Rearranging and summing the above results for all M regional servers from $t = 0, \ldots, T-1$, we have:

$$\sum_{t=0}^{T-1} cE\eta_g\eta_l\mathbb{E}[\nabla f(\mathbf{w}^t)] \leq \sum_{m\in\mathcal{M}}\frac{f(\mathbf{w}_m^0) - F(\mathbf{w}_m^T)}{M} + \sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}}\left(\frac{L\eta_g\eta_l}{2MN_m}\sigma_m^2 + \frac{5N_{m,\theta}EL^2\eta_l^2}{2MN}\beta_{m,\theta}^2\right).$$

This completes the proof. □

A.3 Proof of Theorem 2

Proof. Under partial client participation strategy, we will use $\mathbb{E}_t$ to represent the expectation. Due to the smoothness in Assumption 1, taking expectation of $f(\mathbf{w}^{t+1})$ over the randomness at t-th round as follows:

$$\begin{aligned}
\mathbb{E}[f(\mathbf{w}^{t+1})] \leq f(\mathbf{w}^t) - E\eta_g\eta_l\|\nabla f(\mathbf{w}^t)\|^2 + \eta_g\underbrace{\langle\nabla f(\mathbf{w}^t), \mathbb{E}[\mathbf{w}^{t+1} - \mathbf{w}^t + E\eta_l\nabla f(\mathbf{w}^t)]\rangle}_{B_1} \\
+ \frac{L}{2}\eta_g^2\underbrace{\mathbb{E}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2}_{B_2}.
\end{aligned} \tag{A.2}$$

It is easy to observe that the term B_1 in (A.2) is same as the term A_1 in (A.1) in expectation. Therefore, we only need to focus on the term B_2. For Scheme I under unbiased partial client participation, we assume the sampling subset $S^t = \{I_1, \ldots, I_n, \ldots, I_{K_\theta}\}$ in area type θ, we can

bound the term B_2 as follows:

$$
\begin{aligned}
B_2 &\overset{(1)}{\leq} \frac{E\eta_l^2}{K_\theta}\sigma^2 + \frac{\eta_l^2}{K_\theta^2}\mathbb{E}\Big[\Big\| \sum_{\theta\subseteq 2^{\mathcal{M}}} \sum_{n\in\theta} \sum_{e=0}^{E-1} \nabla F_{l_n}(\mathbf{w}_{l_n}^{t,e})\Big\|^2\Big] \\
&\overset{(2)}{\leq} \frac{E\eta_l^2}{K_\theta}\sigma^2 + \frac{\eta_l^2}{K^2}\sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{K_\theta}{N_\theta}\sum_{i\in\theta}\mathbb{E}\|\mathbf{q}_i\|^2 + \frac{\eta_l^2}{K^2}\sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{K_\theta(K_\theta-1)}{N_\theta^2}\mathbb{E}\Big\|\sum_{i\in\theta}\mathbf{q}_i\Big\|^2,
\end{aligned}
$$

where (1) is based on the Assumption 2 and $\mathbb{E}\|x_1+\cdots+x_n\|^2 \leq n\mathbb{E}(\|x_1\|^2+\cdots+\|x_n\|^2)$; (2) is based on the randomly sampling with replacement and $\mathbf{q}_i = \sum_{e=0}^{E-1}\nabla F_i(\mathbf{w}_{l_n}^{t,e})$. For $\sum_{i\in\theta}\mathbb{E}\|\mathbf{q}_i\|^2$, we have:

$$
\sum_{i\in\theta}\mathbb{E}\|\mathbf{q}_i\|^2 \leq 15N_\theta E^2L^2\eta_l^2G_\theta^2 + (90N_\theta E^4L^2\eta_l^2 + 3N_\theta E^2)\|\nabla f(\mathbf{w}^t)\|^2 + \frac{3E^2N_\theta^2}{N}\alpha_\theta^2. \tag{A.3}
$$

Plugging terms B_1, B_2 and (A.3) into (A.2), if we set the learning rates as $EL\eta_g\eta_l \leq \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{K_\theta(K_\theta-1)}{K^2}$ and the condition $30E^2L^2\eta_l^2 + \frac{L\eta_g\eta_l}{K_m}(90E^3L^2\eta_l^2) < 1$ holding, we have:

$$
\begin{aligned}
\mathbb{E}[f(\mathbf{w}^{t+1})] \leq f(\mathbf{w}^t) - cE\eta_g\eta_l\|\nabla f(\mathbf{w}^t)\|^2 + \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{3N_\theta E^2L\eta_g^2\eta_l^2}{KN^2}\alpha_\theta^2 \\
+ \sum_{\theta\subseteq 2^{\mathcal{M}}}\left(\frac{15N_\theta E^3L^3\eta_g^2\eta_l^4}{KN} + \frac{5N_\theta E^2L^3\eta_g\eta_l^3}{2N}\right)\beta_\theta^2.
\end{aligned}
$$

Rearranging and summing the above results from $t = 0, \ldots, T-1$, we have:

$$
\begin{aligned}
&\sum_{t=0}^{T-1} cE\eta_g\eta_l\mathbb{E}[\nabla f(\mathbf{w}^t)] \leq \sum_{m\in\mathcal{M}}\frac{f(\mathbf{w}_m^0) - F(\mathbf{w}_m^T)}{M} + \sum_{m\in\mathcal{M}}\frac{EL\eta_g^2\eta_l^2}{2MK_m}\sigma_m^2 \\
&+ \sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{3N_{m,\theta}E^2L\eta_g^2\eta_l^2}{MK_mN_m^2}\alpha_{m,\theta}^2 + \sum_{m\in\mathcal{M}}\frac{E\eta_g\eta_l}{M}\sum_{\theta\subseteq 2^{\mathcal{M}}}\left(\frac{L\eta_g\eta_l}{K_mN_m} + \frac{5N_{m,\theta}EL^2\eta_l^2}{2N_m}\right)\beta_\theta^2.
\end{aligned}
$$

For bounding the term B_2 in Scheme II, we have:

$$
\begin{aligned}
B_2 &= \frac{\eta_l^2}{KN}\sum_{\theta\subseteq 2^{\mathcal{M}}}\sum_{i\in\mathcal{N}}\sum_{e=0}^{E-1}\mathbb{E}\Big\|\mathbf{g}_i^{t,e} - \nabla F_i(\mathbf{w}_i^{t,e})\Big\|^2 + \frac{\eta_l^2}{K^2}\mathbb{E}\Big\|\sum_{i\in\theta}\mathbb{I}\{i\in S^t\}\sum_{e=0}^{E-1}\nabla F_i(\mathbf{w}_i^{t,e})\Big\|^2 \\
&= \frac{E\eta_l^2}{K}\sigma^2 + \frac{\eta_l^2}{K^2}\sum_{\theta\subseteq 2^{\mathcal{M}}}\mathbb{E}\Big\|\sum_{i\in\theta}\mathbb{P}\{i\in S_\theta^t\}\mathbf{q}_i\Big\|^2.
\end{aligned}
$$

For bounding $\| \sum_{i\in\theta} \mathbb{P}\{i \in S_\theta^t\}\mathbf{q}_i\|^2$, we have:

$$\begin{aligned}\left\| \sum_{i\in\theta} \mathbb{P}\{i \in S_\theta^t\}\mathbf{q}_i \right\|^2 &= \sum_{i\in\theta} \mathbb{P}\{i \in S_\theta^t\}\|\mathbf{q}_i\|^2 + \sum_{x\neq y} \mathbb{P}\{x, y \in S_{m_i}^t\}\langle \mathbf{q}_x, \mathbf{q}_y\rangle \\ &\quad + \frac{K_\theta^2}{N_\theta}\sum_{i\in\theta}\|\mathbf{q}_i\|^2 - \frac{K_\theta(K_\theta-1)}{2N_\theta(N_\theta-1)}\sum_{x\neq y}\|\mathbf{q}_x - \mathbf{q}_y\|^2.\end{aligned} \tag{A.4}$$

Plugging (A.4) and (A.4) into (A.2), we have:

$$\begin{aligned}&\mathbb{E}[f(\mathbf{w}^{t+1})] \leq \nabla f(\mathbf{w}^t) - E\eta_g\eta_l\left(\frac{1}{2} - 15E^2L^2\eta_l^2\right)||\nabla f(\mathbf{w}^t)||^2 + \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{5N_\theta EL^2\eta_l^3}{2N}\beta_\theta^2 \\ &+ \frac{L\eta_g^2\eta_l^2}{2K^2}\sum_{\theta\subseteq 2^{\mathcal{M}}}\mathbb{E}\left\|\sum_{i\in\theta}\mathbb{P}\{i\in S_\theta^t\}\mathbf{q}_i\right\|^2 - \frac{\eta_g\eta_l}{2EN^2}\sum_{\theta\subseteq 2^{\mathcal{M}}}\mathbb{E}\left\|\sum_{i\in\theta}\mathbf{q}_i\right\|^2 \\ &\leq \nabla f(\mathbf{w}^t) - E\eta_g\eta_l\left(\frac{1}{2} - 15E^2L^2\eta_l^2 - \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{L\eta_g\eta_l(K_\theta-1)}{2K(N_\theta-1)}(90E^3L^2\eta_l^2 + 3E)\right)||\nabla f(\mathbf{w}^t)||^2 \\ &+ \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{2E^2L\eta_g^2\eta_l^2N_\theta(N_\theta-K_\theta)}{KN(N_\theta-1)}\alpha_\theta^2 \\ &+ E\eta_g\eta_l\sum_{\theta\subseteq 2^{\mathcal{M}}}\left(\frac{15N_\theta(N_\theta-K_\theta)E^2L^3\eta_g\eta_l^3}{2NK(N_\theta-1)} + \frac{5N_\theta EL^3\eta_l^2}{2N}\right)\beta_\theta^2 \\ &\leq \nabla f(\mathbf{w}^t) - cE\eta_g\eta_l\|\nabla f(\mathbf{w}^t)\|^2 + \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{2E^2L\eta_g^2\eta_l^2N_\theta(N_\theta-K_\theta)}{KN(N_\theta-1)}\alpha_\theta^2 \\ &+ E\eta_g\eta_l\sum_{\theta\subseteq 2^{\mathcal{M}}}\left(\frac{15N_\theta(N_\theta-K_\theta)E^2L^3\eta_g\eta_l^3}{2NK(N_\theta-1)} + \frac{5N_\theta EL^3\eta_l^2}{2N}\right)\beta_\theta^2.\end{aligned}$$

Rearranging and summing the above results from $t = 0, \ldots, T-1$, we have:

$$\begin{aligned}\sum_{t=0}^{T-1} cE\eta_g\eta_l\mathbb{E}[\nabla f(\mathbf{w}^t)] &\leq \sum_{m\in\mathcal{M}}\frac{f(\mathbf{w}_m^0) - F(\mathbf{w}_m^T)}{M} \\ &+ \sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{2E^2L\eta_g^2\eta_l^2N_{m,\theta}(N_{m,\theta}-K_{m,\theta})}{MK_mN_m(N_{m,\theta}-1)}\alpha_{m,\theta}^2 \\ &+ \frac{E\eta_g\eta_l}{M}\sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{15N_\theta(N_\theta-K_\theta)E^2L^3\eta_g\eta_l^3}{2NK(N_\theta-1)}\beta_{m,\theta}^2 \\ &+ \frac{E\eta_g\eta_l}{M}\sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{5N_\theta EL^3\eta_l^2}{2N}\beta_{m,\theta}^2 + \sum_{m\in\mathcal{M}}\frac{EL\eta_g^2\eta_l^2}{2MK_m}\sigma_m^2.\end{aligned}$$

This completes the proof. □

A.4 Proof of Theorem 3

Proof. For biased partial participation schemes, we can bound $\mathbb{E}[f(\mathbf{w}^{t+1})]$ as same as the results in (A.1). However, we cannot directly leverage the results of B_1 in (A.2), since the result of unbiased sampling in each type area $\mathbb{E}_{\mathcal{K}_m^t}(\mathbf{w}^t) = \frac{1}{N_m}\sum_{i\in\mathcal{N}_m}\mathbf{w}_{i,m}^t$, which is widely used in FL studies [80, 62, 181] is not fit to the biased participation strategy. Therefore, we present a new result the result from the whole regional model to each type area as follows:

$$\mathbb{E}_{\mathcal{K}_{m,\theta}^t}[\mathbf{w}_{m,\theta}] = \mathbb{E}_{\mathcal{K}_{m,\theta}^t}\left[\frac{1}{K_{m,\theta}}\sum_{l\in\mathcal{K}_{m,\theta}^t}\mathbf{w}_{m,l}^t\right] = \frac{1}{K_{m,\theta}}\sum_{l\in\mathcal{K}_{m,\theta}^t}\mathbb{E}_l[\mathbf{w}_{m,l}^t] = \frac{1}{N_{m,\theta}}\sum_{i\in\mathcal{N}_{m,\theta}}\mathbf{w}_{m,i}^t. \tag{A.5}$$

Based on Equation (A.5), we can bound B_1 for the Scheme I of the unbiased partial client participation as follows:

$$\begin{aligned}
B_1 &\leq \frac{E\eta_l}{2}\|\nabla f(\mathbf{w}^t)\|^2 + \frac{E\eta_l}{2K}\sum_{\theta\subseteq 2^{\mathcal{M}}}\sum_{i\in\theta}\mathbb{E}\|(\nabla F_i(\mathbf{w}_i^{t,e}) \\
&- \nabla F_i(\mathbf{w}_i^t))\|^2 - \frac{\eta_l}{2EK^2}\sum_{\theta\subseteq 2^{\mathcal{M}}}\mathbb{E}\left\|\sum_{i\in\theta}\sum_{e=0}^{E-1}\nabla F_i(\mathbf{w}_i^{t,e})\right\|^2 \\
&\leq E\eta_l\left(\frac{1}{2} + 15E^2L^2\eta_l^2\right)\|\nabla f(\mathbf{w}^t)\|^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{5K_\theta E^2L^2\eta_l^3}{2K}\beta_\theta^2 - \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{\eta_l}{2EN_\theta^2}\mathbb{E}\left\|\sum_{i\in\theta}\sum_{e=0}^{E-1}\nabla F_i(\mathbf{w}_i^{t,e})\right\|^2,
\end{aligned} \tag{A.6}$$

For bounding B_2, we have:

$$B_2 \leq \frac{E\eta_l^2}{K}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{\eta_l^2K_\theta}{K^2N_\theta}\sum_{i\in\theta}\|\mathbf{q}_i\|^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{\eta_l^2K_\theta(K_\theta-1)}{K^2N_\theta^2}\left\|\sum_{i\in\theta}\mathbf{q}_i\right\|^2. \tag{A.7}$$

Plugging (A.6) and (A.7) into (A.2), we can bound $\mathbb{E}[f(\mathbf{w}^{t+1})]$ as follows:

$$
\begin{aligned}
\mathbb{E}[f(\mathbf{w}^{t+1})] &\leq \nabla f(\mathbf{w}^t) - E\eta_g\eta_l\left(\frac{1}{2} - 15E^2L^2\eta_l^2\right)\|\nabla f(\mathbf{w}^t)\|^2 \\
&+ \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{5K_\theta E^2L^2\eta_g\eta_l^3}{2K}\left(\sigma^2 + \frac{6EK_\theta}{K}\alpha_\theta^2\right) + \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 \\
&+ \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 + \frac{L\eta_g^2\eta_l^2}{2K}\sum_{\theta\subseteq 2^{\mathcal{M}}}\sum_{i\in\theta}\|\mathbf{q}_i\|^2 \\
&\leq \nabla f(\mathbf{w}^t) - E\eta_g\eta_l\left(\frac{1}{2} - 15E^2L^2\eta_l^2 - \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{L\eta_g\eta_l K_\theta^2}{2K^2N_\theta}(90E^3L^2\eta_l^2 + 3E)\right)\|\nabla f(\mathbf{w}^t)\|^2 \\
&+ \sum_{\theta\subseteq 2^{\mathcal{M}}}\left(\frac{5K_\theta E^2L^2\eta_g\eta_l^3}{2K} + \frac{15K_\theta^2E^2L^3\eta_g^2\eta_l^4}{2K^2N_\theta}\right)\beta_\theta^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{3K_\theta^3E^2L\eta_g^2\eta_l^2}{K^3N_\theta}\alpha_\theta^2 + \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 \\
&\leq \nabla f(\mathbf{w}^t) - cE\eta_g\eta_l\|\nabla f(\mathbf{w}^t)\|^2 + \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 \\
&+ E\eta_g\eta_l\sum_{\theta\subseteq 2^{\mathcal{M}}}\left(\frac{5K_\theta EL^2\eta_l^2}{2K} + \frac{15K_\theta^2E^2L^3\eta_g\eta_l^3}{2K^2N_\theta}\right)\beta_\theta^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{3K_\theta^3E^2L\eta_g^2\eta_l^2}{2K^3N_\theta}\alpha_\theta^2.
\end{aligned}
$$

Rearranging and summing the above results from $t = 0, \ldots, T-1$, we have:

$$
\begin{aligned}
&\sum_{t=0}^{T-1} cE\eta_g\eta_l\mathbb{E}[\nabla f(\mathbf{w}^t)] \leq \sum_{m\in\mathcal{M}}\frac{f(\mathbf{w}_m^0) - F(\mathbf{w}_m^T)}{M} + \sum_{m\in\mathcal{M}}\frac{EL\eta_g^2\eta_l^2}{2MK_m}\sigma_m^2 \\
&+ \sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{3K_{m,\theta}^3E^2L\eta_g^2\eta_l^2}{2MK_m^3N_{m,\theta}}\alpha_{m,\theta}^2 + \sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}}\frac{E\eta_g\eta_l}{M}\left(\frac{5L^2K_{m,\theta}\eta_l^2}{2K_m} + \frac{15K_{m,\theta}^2E^2L^3\eta_g\eta_l^2}{2K_m^2N_{m,\theta}}\right)\beta_{m,\theta}^2.
\end{aligned}
$$

For the Scheme II of biased partial client participation strategy, we have:

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{w}^{t+1})] \leq \nabla f(\mathbf{w}^t) - E\eta_g\eta_l\bigg(\frac{1}{2} - 15E^2L^2\eta_l^2 \\
&- \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{L\eta_g\eta_l K_\theta(N_\theta - K_\theta)}{2KN_\theta(N_\theta-1)}(90E^3L^2\eta_l^2 + 3E)\bigg)\|\nabla f(\mathbf{w}^t)\|^2 \\
&+ \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{3EL\eta_g^2\eta_l^2 K_\theta(N_\theta - K_\theta)}{2K^2N_\theta(N_\theta-1)}\alpha_\theta^2 \\
&+ E\eta_g\eta_l \sum_{\theta\subseteq 2^{\mathcal{M}}}\bigg(\frac{5K_\theta EL^2\eta_l^2}{2K} + \frac{15E^2L^3L\eta_g\eta_l^3 K_\theta(N_\theta - K_\theta)}{2KN_\theta(N_\theta-1)}\bigg)\beta_\theta^2 \\
&\leq \nabla f(\mathbf{w}^t) - cE\eta_g\eta_l\|\nabla f(\mathbf{w}^t)\|^2 + \frac{EL\eta_g^2\eta_l^2}{2K}\sigma^2 + \sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{3EL\eta_g^2\eta_l^2 K_\theta(N_\theta - K_\theta)}{2K^2N_\theta(N_\theta-1)}\alpha_\theta^2 \\
&+ E\eta_g\eta_l \sum_{\theta\subseteq 2^{\mathcal{M}}}\bigg(\frac{5K_\theta EL^2\eta_l^2}{2K} + \frac{15E^2L^3L\eta_g\eta_l^3 K_\theta(N_\theta - K_\theta)}{2KN_\theta(N_\theta-1)}\bigg)\beta_\theta^2.
\end{aligned}
$$

Rearranging and summing the above results from $t = 0, \dots, T-1$, we have:

$$
\begin{aligned}
&\sum_{t=0}^{T-1} cE\eta_g\eta_l\mathbb{E}[\nabla f(\mathbf{w}^t)] \leq \sum_{m\in\mathcal{M}} \frac{f(\mathbf{w}_m^0) - F(\mathbf{w}_m^T)}{M} \\
&+ \sum_{m\in\mathcal{M}} \frac{EL\eta_g^2\eta_l^2}{2MK_m}\sigma_m^2 + \sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{E\eta_g\eta_l}{M}\bigg(\frac{5K_{m,\theta}EL^2\eta_l^2}{2K_m} \\
&+ \frac{15E^2L^3L\eta_g\eta_l^3 K_{m,\theta}(N_{m,\theta} - K_{m,\theta})}{K_mN_{m,\theta}(N_{m,\theta}-1)}\bigg)\beta_{m,\theta}^2 \\
&+ \sum_{m\in\mathcal{M}}\sum_{\theta\subseteq 2^{\mathcal{M}}} \frac{3E^2L\eta_g^2\eta_l^2 K_{m,\theta}(N_{m,\theta} - K_{m,\theta})}{2MK_m^2N_{m,\theta}(N_{m,\theta}-1)}\alpha_{m,\theta}^2.
\end{aligned}
$$

This completes the proof. □

Appendix B: Proofs of Chapter 4

In this appendix, we present all lemma and theorems in Chapter 4 in detail.

B.1 Explanation of Definition 2

Proof. The i-th remote update $\mathbf{u}_i$ comes from the local training process on the private dataset $\mathcal{D}_i$, which could be denoted as a set of training sample pairs $\mathcal{D}_i = \{\mathbf{x}_i, y_i\}$ where $\mathbf{x}_i$ is the training dataset input vector and y_i is the output result. Additionally, we can describe the relationship between predicting output vector from the learning classifier (donated as $\hat{y}_i$) and the input matrix $\mathbf{x}_i$ as $\hat{y}_i = softmax(\mathbf{u}_i^T \times \mathbf{x}_i + b)$, where b is the bias of learning model. As the learning update $\mathbf{u}_i$ is high correlated to the training data distribution, (which is also assumed in [139]), we assume that $\mathbf{u}_i$ follows a similar distribution with $\mathbf{x}$. Then the result probability of y at a certain output label $r \in [1, 2, ...R]$ can be written as $\mathbb{P}(y_r) = \mathbb{P}(y_r|\mathcal{U})$, where $\mathcal{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{N+M}\}$. However, in our defender, the distribution of $\mathbf{u}_i$ is not known to the defender and we could estimate the probability distribution of $\mathbf{y}$ according to each label. Specially, assuming that the contribution from the parameters in $\mathbf{u}_i$ are independent for each output label as $\mathbb{P}(\mathbf{u}_i^r, \mathbf{u}_i^{r'}|y_r) = \mathbb{P}(\mathbf{u}_i^r|y_r)\mathbb{P}(\mathbf{u}_i^{r'}|y_r)$ when $r \neq r'$, then we can estimate the density distribution of $\mathbf{u}_i$ from the distribution of $\mathbf{y}$ as

$$\begin{aligned}\mathbb{P}(\mathbf{u}_i) &= \mathbb{P}((\mathbf{u}_i^1, \mathbf{u}_i^2, ..., \mathbf{u}_i^R)|y_r)\mathbb{P}(y_r) \\ &= \prod_{r=1}^{R} \mathbb{P}(\mathbf{u}_i^r|y_r)\mathbb{P}(y_r) = \prod_{r=1}^{R} \mathbb{P}(\mathbf{u}_i^r|y)\mathbb{P}(y).\end{aligned}$$

Then the value of $F(i)^r$ for r-th label can be written as

$$F(i)^r = \frac{\sum_{j=1}^{k} \tilde{q}(\mathbf{u}_j^r)}{k\tilde{q}(\mathbf{u}_i^r)} = \frac{\sum_{j=1}^{k} \tilde{q}(\mathbf{u}_j^r|y)|\mathbb{P}(y)}{k\tilde{q}(\mathbf{u}_i^r|y)\mathbb{P}(y)} \tag{B.1}$$

Taking this equation into the definition of $F(i)$ as

$$F(i) = \frac{\sum_{j=1}^{k} \tilde{q}(\mathbf{u}_j)}{k\tilde{q}(\mathbf{u}_i)} = \frac{\sum_{j=1}^{k} \prod_{r=1}^{R} \tilde{q}(\mathbf{u}_j^r|y)\mathbb{P}(y)}{k\tilde{q}\prod_{r=1}^{R}(\mathbf{u}^r|y)\mathbb{P}(y)} = \prod_{r=1}^{R} F(i)^r.$$

This completes the proof. □

B.2 Proof of Theorem 8

Proof. The density estimation at ith update $\mathbf{u}_i$ is

$$q(\mathbf{u}_i) = \frac{1}{k+1}\sum_{j=1}^{k} \frac{1}{(2\pi)^{r/2}h^r} \exp\left(-\frac{|\mathbf{u}_{i,j}|^2}{2\bar{h}}\right) \tag{B.2}$$

and similar in [149] the average local density estimation in the neighborhood of $\mathbf{u}_i$ could be defined as

$$\bar{q}(\mathbf{u}_i) = \frac{1}{k}\sum_{j=1}^{k} q(\mathbf{u}_j) = \frac{1}{k(k+1)}\sum_{j=1}^{k} \frac{1}{(2\pi)^{\frac{r}{2}}h^r} \exp\left(-\frac{|\mathbf{u}_i - \mathbf{u}_{i,j}|^2}{2\bar{h}}\right) \tag{B.3}$$

Assuming that $\mathbf{u}_{i,j}$ is uniformly distributed in a o dimension ball B_β, where β is the radius of the ball (i.e., the distance between kth nearest neighborhood and $\mathbf{u}_i$) we can calculate the expectation of $q(\mathbf{u}_i)$ and $\bar{q}(\mathbf{u}_i)$

$$\mathbb{E}(\bar{q}(\mathbf{u}_i)) = \mathbb{E}(q(\mathbf{u}_i)) = \frac{1}{V} = \frac{2\pi^{(r-1)/2}\beta^{r-1}}{\Gamma((r-1)/2+1)} \tag{B.4}$$

where V is the volume of B_β. The rest of proof is followed by the McDiarmid's Inequality [105] which gives the upper bound of the probability that a function of $g(\mathbf{u}_{i,1}, \dots, \mathbf{u}_{i,j}, \dots, \mathbf{u}_{i,k})$ deviates with the expectation. Let $g : \mathbb{R}^r \to \mathbb{R}, \forall j, \forall \mathbf{u}_{i,1}, \mathbf{u}_{i,k}, \mathbf{u}'_{i,j} \in \mathcal{U}_i$, we have

$$|g(\mathbf{u}_{i,1}, \dots, \mathbf{u}_{i,j}, \dots, _{i,k}) - g(\mathbf{u}_{i,1}, \dots, \mathbf{u}'_{i,j}, \dots, \mathbf{u}_{i,k})| \leq \xi_{i,j} \tag{B.5}$$

For all $\gamma > 0$, we have $\mathbb{P}(g - \mathbb{E}(g) \geq \gamma) \leq \exp\left(\frac{-2\gamma^2}{\sum_{j=1}^{k} c_{i,j}^2}\right)$. For $g_1 = q(\mathbf{u}_i)$,

$$\begin{aligned} &|g_1(\mathbf{u}_{i,1}, \dots, \mathbf{u}_{i,j}, \dots, \mathbf{u}_{i,k}) - g_1(\mathbf{u}_{i,1}, \dots, \mathbf{u}'_{i,j}, \dots, \mathbf{u}_{i,k})| \\ &= \frac{K(\mathbf{u}_{i,j}/\bar{h}) - K(\mathbf{u}'_{i,j}/\bar{h})}{\bar{h}^r(k+1)} \leq \frac{1 - \exp(-\beta^2/2\bar{h})}{(2\pi)^{r/2}\bar{h}^r(k+1)} = \xi_1 \end{aligned} \tag{B.6}$$

For $g_2 = \bar{q}(\mathbf{u}_i)$,

$$\begin{aligned} &|g_2(\mathbf{u}_{i,1}, \dots, \mathbf{u}_{i,j}, \dots, \mathbf{u}_{i,k}) - \mathbf{u}_2(\mathbf{u}_{i,1}, \dots, \mathbf{u}'_{i,j}, \dots, \mathbf{u}_{i,k})| \\ &= \frac{K\left(\frac{\mathbf{u}_{i,j}}{\bar{h}}\right) - \left(\frac{\mathbf{u}'_{i,j}}{\bar{h}}\right) + 2\sum_{j=1}^{k}\left[K\left(\frac{\mathbf{u}_i - \mathbf{u}_{i,j}}{\bar{h}}\right) - \left(\frac{\mathbf{u}'_i - \mathbf{u}_{i,j}}{\bar{h}}\right)\right]}{\bar{h}^r(k+1)} \\ &\leq \frac{1 - \exp(-\beta^2/2\bar{k}) + 2k(1 - \exp(-2\beta^2/\bar{h}))}{(2\pi)^{r/2}\bar{h}^r(k+1)} = \xi_2 \end{aligned} \tag{B.7}$$

The probability of LMF is

$$\mathbb{P}(LMF(i) > \epsilon_m) = \mathbb{P}[\bar{q}(\mathbf{u}_i - \epsilon_m q(\mathbf{u}_i))] = \mathbb{P}(g - \mathbb{E}(g) > \theta), \tag{B.8}$$

where $\theta = (\epsilon_m - 1)/V$. We are only interested in the case of $LMF(i) > 1$, i.e., $\epsilon_m > 1$, and $\theta > 0$. Following by the McDiarmid's Inequality, we have

$$\begin{aligned} &\mathbb{P}(LMF(i) > \epsilon_m) \leq \exp\left(-\frac{2\theta^2}{\sum_{j=1}^{k}\xi^2}\right) = \exp\left(-\frac{2\theta^2}{k\xi^2}\right) \\ &\leq \exp\left(-\frac{2(\epsilon_m - 1)^2(k+1)^2(2\pi)^r\bar{h}^{2r}}{k(2k+\epsilon+1)^2V^2}\right) \end{aligned} \tag{B.9}$$

This completes the proof. □

Appendix C: Proofs of Chapter 5

In this appendix, we present all theorems in Chapter 5 in detail.

C.1 Proof of Theorem 10

Because we cannot know the correlation coefficient exactly, we need to consider the positive and negative correlation simultaneously. Let θ be the transition probability on the generation tree, and we assume $\theta < 0.5$. In [87], they proved that $\theta = \frac{1-\rho}{2}$. Because Eve cannot know the bit sequence is positive correlation or not, she needs to search from the most likely bit sequences happen with probabilities θ^L and $(1-\theta)^L$ simultaneously. MLTS searches for X compute $K_T = f_c(X)$ from the most likely bit sequence towards the least likely one in $\{0,1\}^L$. Given the fact K_D has been established, the MLTS success probability searching for K_D is

$$\begin{aligned}
&\mathbb{P}(\text{MLTS succeeds} \mid K_D \text{ established}) \\
&= \sum_{i=0}^{n/2} \binom{L}{i} \theta^i (1-\theta)^{L-i} + \sum_{i=0}^{n/2} \binom{L}{i} (1-\theta)^i \theta^{L-i} \\
&= I_{\frac{1-\rho}{2}}\left(L - \frac{n}{2}, \frac{n}{2} + 1\right) + I_{\frac{1+\rho}{2}}\left(L - \frac{n}{2}, \frac{n}{2} + 1\right), \qquad \text{(C.1)}
\end{aligned}$$

where $0 \leq n \leq L$, and $I_x(a,b)$ is the regularized incomplete beta function that has been defined previously. □

C.2 MLTS for Multi-level Quantization

If we use m levels to quantify the wireless information, each level can be represented by a b-bit string, where $m = \log_2 b$. The multi-level quantization is also called m-ary quantization. Here, we redefine the correlation coefficient ρ_m of consecutive bit arrays x_i^m and x_{i+1}^m, where

$x_i^m = \{x_{i,1}^m, \ldots, x_{i,m}^m\}$ in bit sequence X as follows

$$\rho_m = \frac{\sum_{j=1}^{m}(x_{i,j}^m - \bar{x}_i^m)(x_{i+1,j}^m - \bar{x}_{i+1}^m)}{\sqrt{\sum_{j=1}^{m}(x_{i,j}^m - \bar{x}_i^m)^2}\sqrt{\sum_{j=1}^{m}(x_{i+1,j}^m - \bar{x}_{i+1}^m)^2}}. \tag{C.2}$$

For the b-ary quantization, transition probability $\theta_{r,s}$ (i.e., $\mathbb{P}(x_{i+1}^m = s | x_i^m = r)$), where r and s are the states, $r, s \in \{1, 2, \ldots, m\}$ in transition matrix $\Phi \in \mathbb{R}^{m \times m}$ as $\theta_{r,s} = \rho\delta_{r,s} + (1-\rho)/2^m$, where $\delta_{r,s}$ is Kronecker delta [164]. m-ary searching is equivalent to dividing the L-bit sequence to 2^{m-1} blocks and the number of possible initialization is $\frac{n}{2^m}$. Thus, $\mathbb{P}(\text{MLTS succeeds} \mid K_D \text{ established})$ is given as

$$\begin{aligned}
&\mathbb{P}(\text{MLTS succeeds} \mid K_D \text{ established}) \\
&= \binom{\frac{L}{2^{m-1}}}{0} \theta_{1,1}^{\frac{L}{2^{m-1}}} + \cdots + \binom{\frac{L}{2^{m-1}}}{1} \theta_{1,1}^{\frac{L}{2^{m-1}}-1} \theta_{1,2} + \cdots \\
&+ \binom{\frac{L}{2^{m-1}}}{1} \theta_{1,1}^{\frac{L}{2^{m-1}}-1} \theta_{1m} + \cdots + \binom{\frac{L}{2^{m-1}}}{\frac{n}{2^m}} \theta_{1,1}^{\frac{L}{2^{m-1}}-\frac{n}{2^m}} \theta_{1,2}^{\frac{n}{2^m}} \\
&+ \cdots + \binom{\frac{L}{2^{m-1}}}{\frac{n}{2^m}} \theta_{1,1}^{\frac{L}{2^{m-1}}-\frac{n}{2^m}} \theta_{1,2}^{\frac{n}{2^m}} + \cdots \\
&= \sum_{s=1}^{m} \sum_{i=0}^{\frac{n}{2^m}} \binom{\frac{L}{2^{m-1}}}{i} \theta_{s,s}^{\frac{L}{2^{m-1}}-i} \theta_{s,\neg s}^{i} = \sum_{s=1}^{m} I_{\rho+\frac{1-\rho}{2^m}} \left(\frac{L}{2^{m-1}} - \frac{n}{2^m}, \frac{n}{2^m} + 1 \right),
\end{aligned}$$

where $\theta_{s,\neg s}$ is the transition probability from state s to the rest of states except s.

C.3 Theoretical Results of NIST Randomness Tests

Due to the page limitation, we give the results of other 7 different tests and ignore the proof. From NIST test suite [14], we can conclude that the P-value can be calculated by Gaussian distribution: frequency test (Frequency), run test (Run) and DFT test (DFT) and Chi-square distribution: frequency test within a block test (BlockFreq), longest run of ones in a block test (LongRun), non-overlapping template matching test (Nonoverlap), approximate entropy test (AppEntropy), first order serial (1storder) and second order serial (2ndorder).

For Run test, $\mathbb{P}(T_{\text{run}} \text{ accepts } H_0) = h_{\text{run}}(\rho, \alpha) = \frac{1}{2}\text{erf}\left(\frac{\alpha_f + \mu_f}{\sigma_f\sqrt{2}}\right) + \frac{1}{2}\text{erf}\left(\frac{\alpha_f - \mu_f}{\sigma_f\sqrt{2}}\right)$, where $\alpha_f = 2\sqrt{2L}\pi(1-\pi)\text{erfc}^{-1}(\alpha)$, $\mu_f = L\lambda - 2L(1-\pi)$ and $\sigma_f = \sqrt{L\lambda(1-\lambda)}$. The definition

of π is the probability of I_l, and λ is the meaning of $I_l = 1$ if the lth element $\neq$ the (l-1)th element; $I_l = 0$ otherwise.

For DFT test, $\mathbb{P}(T_{\text{DFT}}$ accepts $H_0) = h_{\text{DFT}}(\rho, \alpha) = \frac{1}{2}\text{erf}\left(\frac{\alpha_f + \mu_f}{\sigma_f\sqrt{2}}\right) + \frac{1}{2}\text{erf}\left(\frac{\alpha_f - \mu_f}{\sigma_f\sqrt{2}}\right)$, where $\alpha_f = \text{erfc}^{-1}(\alpha)\sqrt{\mathbb{P}_{\text{DFT}}(1-\mathbb{P}_{\text{DFT}})\frac{n}{4}}$, $\mu_f = 0.95\frac{n}{2} - \mathbb{P}_{\text{DFT}}\frac{n}{2}$, and $\sigma_f = \sqrt{(0.95)(0.05)\frac{n}{4}}$. Accordingly, $\mathbb{P}_{\text{DFT}} = \mathbb{P}(n|S_j(R)| < -n\ln(0.05))$, where $S_j(R)$ is defined in NIST.

For BlockFreq test, The statistic is $\chi^2(\text{obs}) = 4M\sum_{i=1}^{N}(\pi_i - \frac{1}{2})^2$. Similar to the result of Frequency, $\pi_i \sim \mathcal{N}(\frac{1}{2}, \frac{4(1+\rho)}{1-\rho})$. The probability is $\mathbb{P}(T_{\text{longestrun}}$ accepts $H_0) = h_{\text{longestrun}}(\rho, \alpha) = \text{igam}\left(\frac{N}{2}, \frac{2\text{igamc}^{-1}(N/2,\alpha)\cdot\chi^2(\text{newobs})}{\chi^2(\text{obs})}\right) - \text{igam}\left(\frac{N}{2}, \chi^2(\text{newobs})\right)$, where igam is incomplete gamma integral function and igamc^{-1} is inverse complemented incomplete gamma integral function. obs is the statistic by calculating the number and newobs is calculated by the exact distribution $\mathbb{P}(\pi_i)$.

For LongRun test, The statistic is $\chi^2(\text{obs}) = \sum_{i=0}^{K}\frac{(v_i - N\pi_i)^2}{N\pi_i}$, where π_i is the statistical probability of obs for i, K and N are defined in NIST test suite. The exact distribution of v_i is $\mathbb{P}(v_i) = \xi_0\mathbf{M}^i\mathbf{1}^T$, where ξ_0 is the initial $[1/2, 1/2, 0, \ldots, 0]_{1\times(i+1)}$, $\mathbf{1}^T$ is the transpose of the row vector $\mathbf{1} = [1, 1, \ldots, 1]_{1\times(i+1)}$, and the $(i+1)\times(1+1)$ matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 \\ 0 & \mathbb{P}_{00} & \mathbb{P}_{01} & 0 & \cdots & 0 \\ 0 & \mathbb{P}_{10} & 0 & \mathbb{P}_{11} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \mathbb{P}_{10} & 0 & 0 & \cdots & \mathbb{P}_{11} \\ 0 & \mathbb{P}_{10} & 0 & 0 & \cdots & 0 \end{bmatrix}$$

where $\mathbb{P}_{00} = \mathbb{P}_{11} = 1/2 + \rho/2$ and $\mathbb{P}_{01} = \mathbb{P}_{10} = 1/2 - \rho/2$. The probability of longestrun is $\mathbb{P}(T_{\text{longestrun}}$ accepts $H_0) = h_{\text{longestrun}}(\rho, \alpha) = \text{igam}\left(\frac{K}{2}, \frac{2\text{igamc}^{-1}(K/2,\alpha)\cdot\chi^2(\text{newobs})}{\chi^2(\text{obs})}\right)$
$-\text{igam}\left(\frac{K}{2}, \chi^2(\text{newobs})\right)$.

For Nonoverlap test, the statistic is $\chi^2(\text{obs}) = \sum_{i=1}^{N}\frac{(W_i - \mu)^2}{\sigma^2}$, where $\mu = (M - m + 1)/2^m$ and $\sigma^2 = M\left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right)$. Now, we will compute the exact distribution of W_i. Let W_i be a template with binary numbers from $\mathcal{A} = \{0, 1\}$, and define the indicator variable $I_a(W_i)$

of the appearance of W_i at the position a is $I_a(W_i) = I[X_{a-m+1} = a_1, \dots, X_a = a_m]$ with the expectation $\eta = \pi(a_1)\prod_{t=1}^{m-1}\mathbb{P}(a_t, a_{t+1})$. $\beta = (M - m + 1)y$, where y is the solution of equation $ye = \eta$. Because the Markovian hypothesis and the combinatorial structure of the problem requires some more notations in addition, $e = 1 + \sum_{t=1}^{m-1}\epsilon(t)C(t)$, where $\epsilon(t) = 1$, if there is an overlap of length t two Ws; $\epsilon(t) = 0$, otherwise. $C(t) = \mathbb{P}(a_m, a_{t+1})$, if $t = m-1$; $\mathbb{P}(a_m, a_{t+1})\prod_{l=t+1}^{m-1}\mathbb{P}(a_l, a_{l+1})$, if $t < m-1$.

The quantity $C(t)$ can be thought of as being the probability of observing the $m - t$ last letters of W successively. Now, W_i follows by Poisson distribution $W_i \sim Po(\eta)$. Thus, $\mathbb{P}(T_{\text{nonoverlapping}}$ accepts $H_0) = h_{\text{nonoverlapping}}(\rho, \alpha) =$
$\text{igam}\left(\frac{N}{2}, \frac{2\text{igamc}^{-1}(K/2,\alpha)\cdot\chi^2(\text{newobs})}{\chi^2(\text{obs})}\right) - \text{igam}\left(\frac{N}{2}, \chi^2(\text{newobs})\right)$.

For AppEntropy test, the statistic is $\chi^2(\text{obs}) = 2N[\log 2 - ApEn(m)]$, where $ApEn = \phi^{(m)} - \phi^{(m+1)}$, m is the overlapping block size and $\phi^{(m)} = \sum_{i=1}^{2^m}\frac{\text{freq of block } i}{N}\times\log(\frac{\text{freq of block } i}{N})$. Based on Markov dependent random variables, we define $\mathbb{P}_i^{(m)} = \mathbb{P}(U_{i_1})\mathbb{P}(U_{i_2}|U_{i_1})\cdot\mathbb{P}(U_{i_m}|U_{i_{m-1}})$, where the conditional probability can be calculated by transition matrix with ρ in [87]. Now, we can define the new AppEntropy as $\chi^2(\text{newobs}) = 2N[\log 2 - \tilde{\phi^{(m)}} + \tilde{\phi^{(m+1)}}]$, where $\tilde{\phi^{(m)}} = \sum_{i=1}^{2^m}\mathbb{P}_i^{(m)}\log\mathbb{P}_i^{(m)}$. Thus, we obtain that $\mathbb{P}(T_{\text{AppEntropy}}$ accepts $H_0) = h_{\text{AppEntropy}}(\rho, \alpha) =$ $\text{igam}\left(2^{m-1}, \frac{\text{igamc}^{-1}\left(2^{m-1},\alpha\right)\cdot\chi^2(\text{newobs})}{chi^2(\text{obs})}\right) - \text{igam}\left(2^{m-1}, \frac{\chi^2(\text{newobs})}{2}\right)$.

For 1storder test, v_{i_m}, $v_{i_{m-1}}$ and $v_{i_{m-2}}$ denotes the m-bit, $m-1$-bit and $m-2$-bit matching pattern. U is each random variable from source emitting in one matching block. For Markov dependent random variables, we can obtain each matching probability of different matching pattern $\mathbb{P}_{i_m}(v_{i_m} = U_j) = \mathbb{P}(U_1)\times\mathbb{P}(U_2|U_1)\times\cdots\times\mathbb{P}(U_m|U_{m-1})$, and the expectation is $\mathbb{E}_{i_m}(v_{i_m} = U_j) = n\times\mathbb{P}(U_1)\times\cdots\times\mathbb{P}(U_m|U_{m-1})$. The statistic of $\psi_m^2 = \sum_{i_m=1}^{2^m}\frac{(v_{i_m}-\mathbb{E}_{i_m})^2}{\mathbb{E}_{i_m}}$. Then, we compute $\Delta\hat{\psi}_m^2 = \psi_m^2 - \psi_{m-1}^2$ and $\Delta^2\hat{\psi}_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2$. The probability of first order serial is $\mathbb{P}(T_{1\text{storder}}$ accepts $H_0) = h_{1\text{storder}}(\rho, \alpha) = \text{igam}\left(2^{m-2}, \frac{2\text{igamc}^{-1}(2^{m-2},\alpha)\cdot\Delta\hat{\psi}_m^2}{\Delta\psi_m^2}\right) -$ $\text{igam}\left(2^{m-2}, \Delta\hat{\psi}_m^2\right)$.

For 2ndorder test, we can obtain that $\mathbb{P}(T_{2\text{ndorder}}$ accepts $H_0) = h_{2\text{ndorder}}(\rho, \alpha) =$
$\text{igam}\left(2^{m-3}, \frac{2\text{igamc}^{-1}(2^{m-3},\alpha)\cdot\Delta^2\hat{\psi}_m^2}{\Delta^2\psi_m^2}\right) - \text{igam}\left(2^{m-3}, \Delta^2\hat{\psi}_m^2\right)$.

Appendix D: Copyright Permissions

This is the permission for the reuse of contents in Chapter 2.

9/6/22, 4:15 PM Rightslink® by Copyright Clearance Center

RightsLink

Home | Help | Live Chat | Zhe Qu

IEEE Requesting permission to reuse content from an IEEE publication

On the Convergence of Multi-Server Federated Learning with Overlapping Area

Author: Zhe Qu; Xingyu Li; Jie Xu; Bo Tang; Zhuo Lu; Yao Liu
Publication: IEEE Transactions on Mobile Computing
Publisher: IEEE
Date: 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

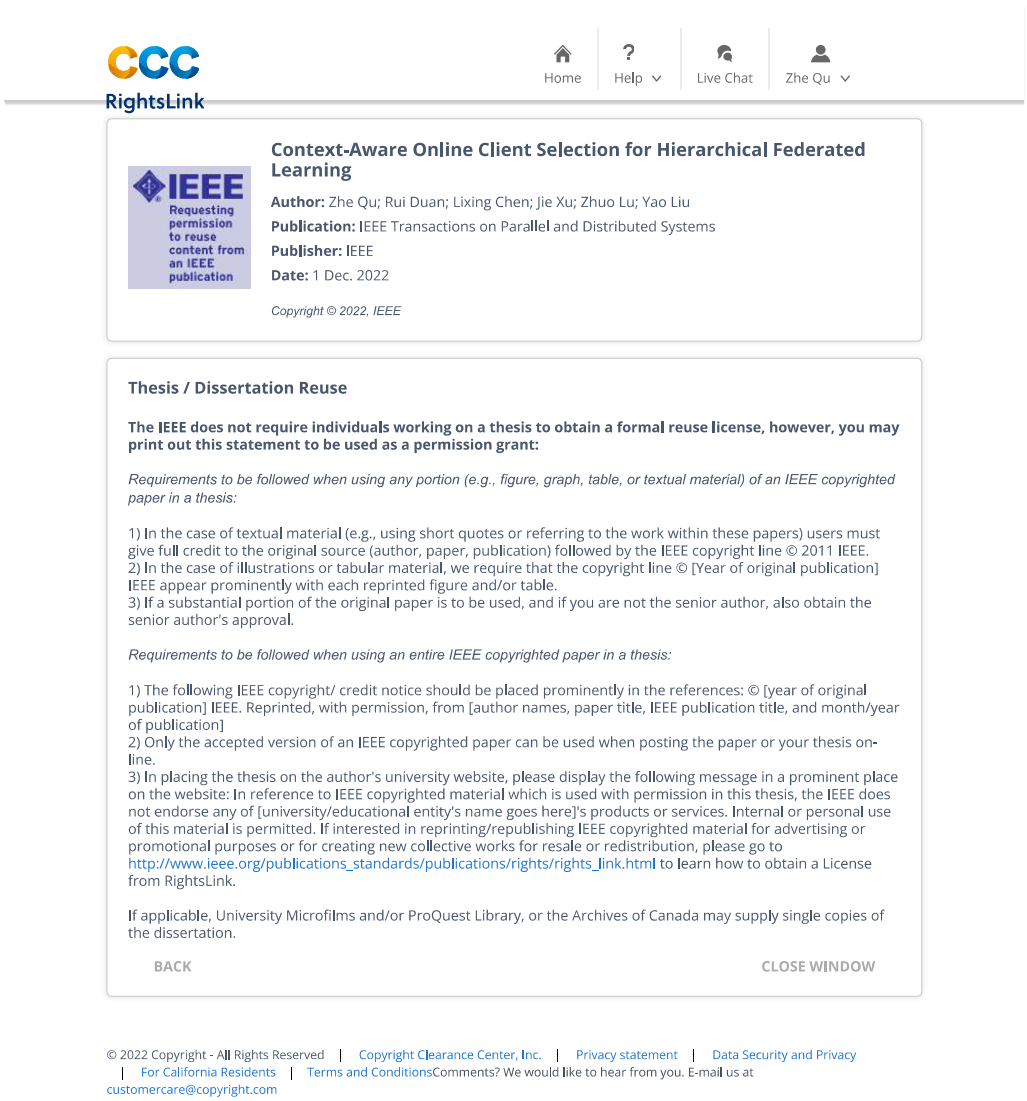
Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

This is the permission for the reuse of contents in Chapter 3.



RightsLink

Home | Help | Live Chat | Zhe Qu

IEEE Requesting permission to reuse content from an IEEE publication

Context-Aware Online Client Selection for Hierarchical Federated Learning

Author: Zhe Qu; Rui Duan; Lixing Chen; Jie Xu; Zhuo Lu; Yao Liu
Publication: IEEE Transactions on Parallel and Distributed Systems
Publisher: IEEE
Date: 1 Dec. 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

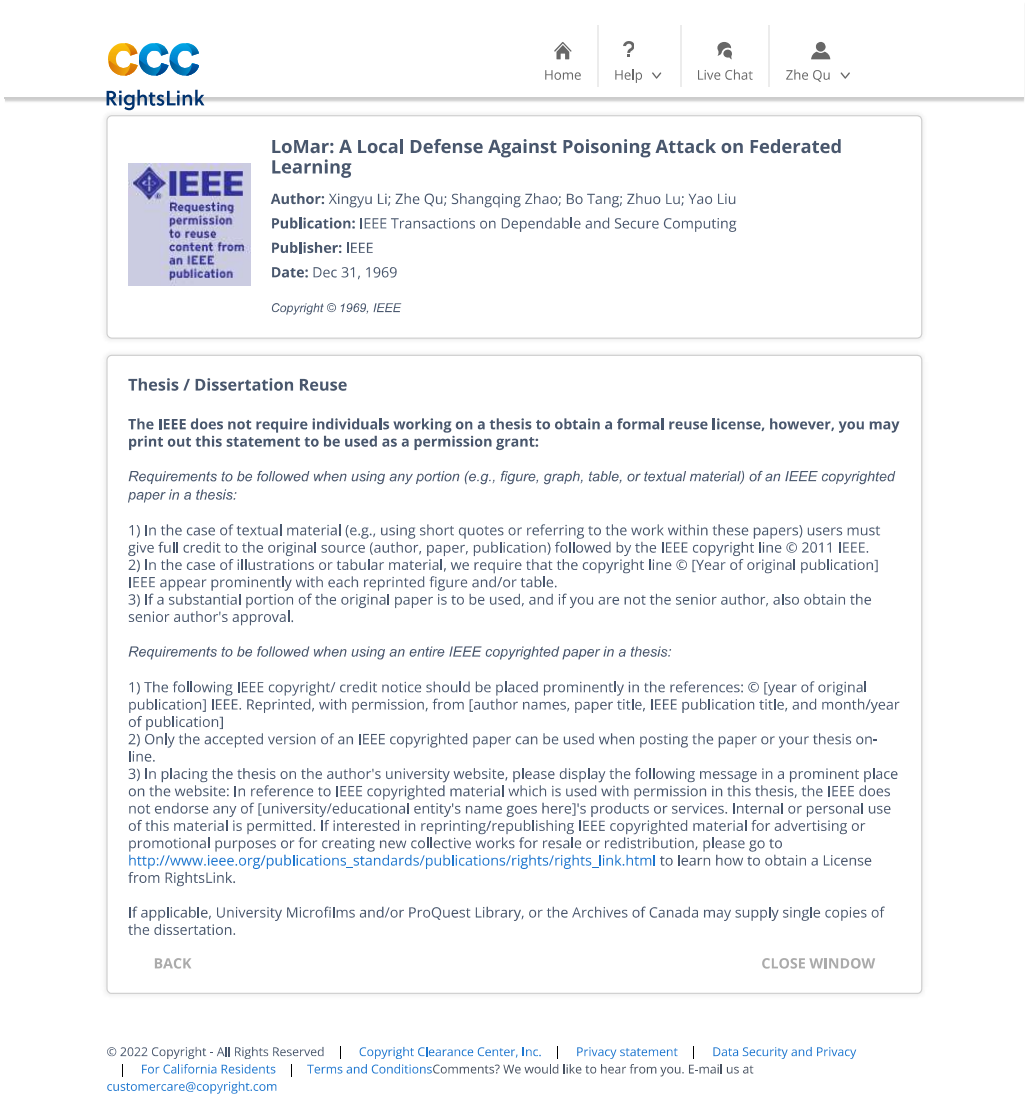
Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

This is the permission for the reuse of contents in Chapter 4.



RightsLink

Home | Help | Live Chat | Zhe Qu

IEEE Requesting permission to reuse content from an IEEE publication

LoMar: A Local Defense Against Poisoning Attack on Federated Learning

Author: Xingyu Li; Zhe Qu; Shangqing Zhao; Bo Tang; Zhuo Lu; Yao Liu
Publication: IEEE Transactions on Dependable and Secure Computing
Publisher: IEEE
Date: Dec 31, 1969

Copyright © 1969, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

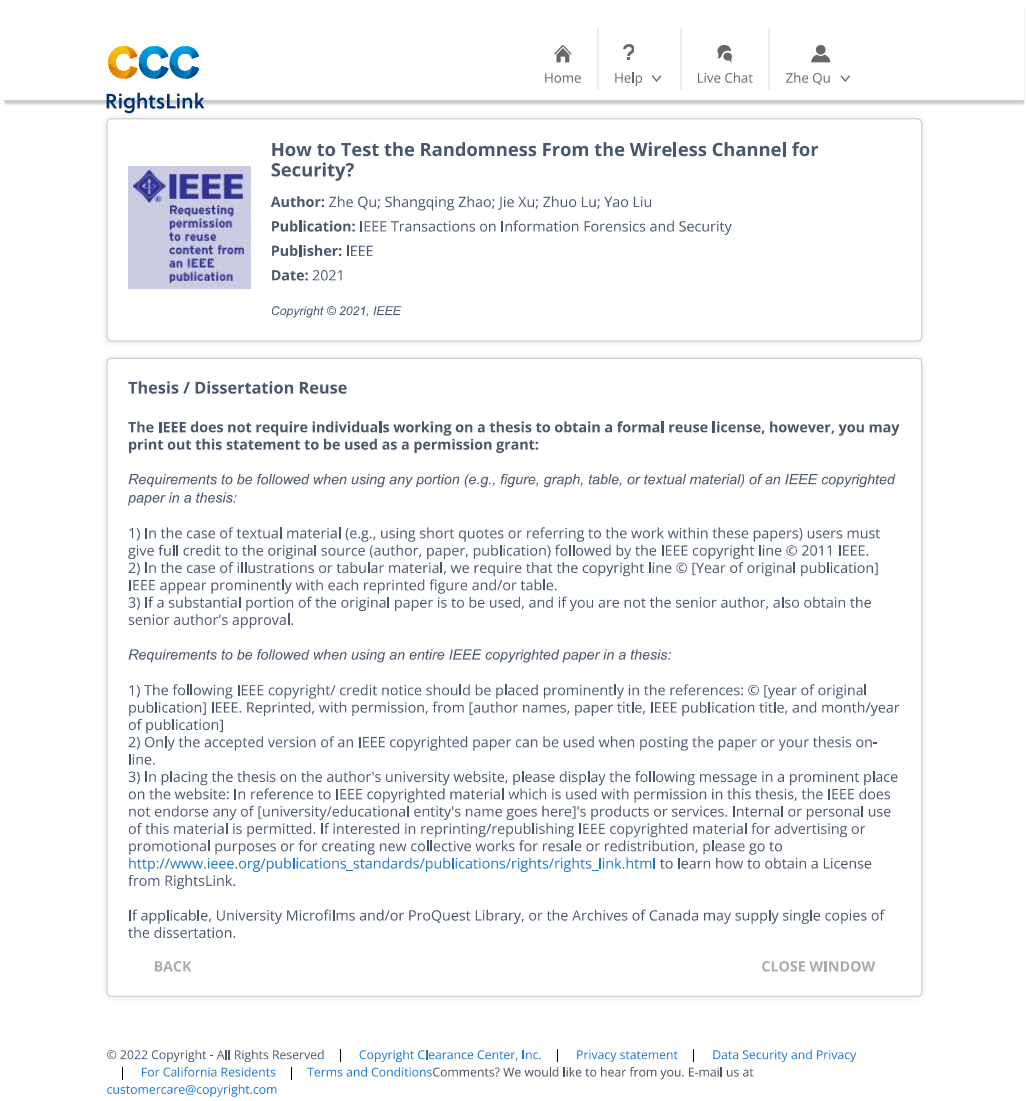
Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

This is the permission for the reuse of contents in Chapter 5.



RightsLink

Home | Help | Live Chat | Zhe Qu

How to Test the Randomness From the Wireless Channel for Security?

Author: Zhe Qu; Shangqing Zhao; Jie Xu; Zhuo Lu; Yao Liu
Publication: IEEE Transactions on Information Forensics and Security
Publisher: IEEE
Date: 2021

Copyright © 2021, IEEE

IEEE Requesting permission to reuse content from an IEEE publication

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW