Key Generation and Secure Coding in Communications and Private Learning

by

Nasser Aldaghri

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan
2022

Doctoral Committee:

Professor Hessam Mahdavifar, Chair
Professor Mosharaf Chowdhury
Professor S. Sandeep Pradhan
Professor Wayne E. Stark

Nasser Aldaghri
aldaghri@umich.edu
ORCID iD: 0000-0003-4308-8631

DEDICATION

To my beloved parents and siblings.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my God for giving me the opportunity, will, and strength to complete my Ph.D. journey. Second, I would like to express my gratitude to my mentor and advisor, Prof. Hessam Mahdavifar, for his knowledge, patience, and continuous support throughout my doctoral studies. My research outcomes would not be feasible without his help and mentorship. I would also like to thank Dr. Ahmad Beirami for his time and help in the latter years of my Ph.D. Additionally, I would like to thank Prof. Chowdhury, Prof. Pradhan, and Prof. Stark for serving on my dissertation committee.

I would like to express my gratitude to King Saud University in Riyadh, Saudi Arabia, for providing me with the financial support needed to complete my graduate studies. I would also like to thank all my friends who I met in Ann Arbor, their company has brought me a lot of joy, happiness, and support whenever I needed.

Finally, I would like to extend my gratitude to my family. I would not be able to complete this journey without the endless love, encouragement, and support I have received from my parents Abdulrahman and Aljawharah. There are no words that can express my sincere gratitude and appreciation for them. I would also like to thank my siblings, who have always been supportive and provided me with advice and encouragement throughout my life.

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

ABSTRACT

The increasingly distributed nature of many current and future technologies has introduced many challenges for devices designed for such settings. Devices operating in such environments, such as Internet-of-Things (IoT), medical devices, connected vehicles, etc., typically have limited computational power and rely on batteries to operate. Therefore, efficiency is a paramount requirement for any algorithm designed to be implemented on these devices. Furthermore, these devices typically generate and collect huge amounts of extremely sensitive and personal data, such as health-related data, behavior-related data, etc. As a result, there is a need for security and privacy protections to guard against various attacks. Additionally, since these devices are typically resource-constrained, any algorithm or protocol needs to be efficient to enable its implementation on such devices. Efficient security and privacy solutions are essential to cope with, as well as enable, high deployment rate of such devices for various sensitive applications.

In this dissertation, efficient solutions for protecting the security and privacy of data generated by such devices are explored. Low-complexity protocols for generating secret keys in static environments, along with a formulation of threshold-secure coding with a shared key and corresponding coding schemes are presented. Additionally, algorithms for coded machine unlearning for regression problems are presented, as well as a new setup and algorithm for federated learning with opt-out differential privacy are presented and evaluated.

CHAPTER 1

Introduction

1.1 Growth of Edge Devices

The number of wireless edge devices in communication networks has seen incredible growth over the past decade with no slowdown in sight. This rapid growth is driven by advancements in communication protocols, algorithms, and electronics. These devices are and will be playing a bigger part in future technologies, such as artificial intelligence applications. Examples of edge devices include Internet-of-Things (IoT) devices, medical devices, connected vehicles, etc. In fact, IoT devices overtook non-IoT devices in terms of active connections globally in 2020 and will be triple the number of non-IoT active connections by the year 2025 [5]. Furthermore, in addition to the aforementioned devices, personal devices, such as phones, are adopting direct communication technologies such as device-to-device (D2D) communications in fifth-generation (5G) wireless network communication for mobile devices. These technologies, and many others, enable a massive number of devices to communicate directly rather than communicating through the core network, presenting many new challenges to existing solutions.

The data generated by connected devices typically include personal, and often extremely sensitive, information. For example, a smartwatch may collect multiple personal health metrics, and a smart home sensor may collect patterns of movements of residents. IDC estimates that connected devices will generate 79.4 zettabytes of data by the year 2025, up from 13.6 zettabytes in 2019 [6].

One of the common characteristics of these devices is their reliance on batteries and limited computational power. These limitations dictate the requirement of any algorithm intended for deployment on such devices to be efficient.

The rapid increase in the generation of sensitive data on connected devices and the low-complexity requirement of algorithms highlight the need for efficient algorithms to preserve the security and privacy of such data. These challenges have motivated a lot of research efforts to address such issues and have been an active area of interdisciplinary research over the past decade.

1.2 Key Generation and Secure Coding

Data security has been an instrumental part of communication systems. Generally, the formulation of secrecy problems is as follows. There are two legitimate parties referred to as Alice and Bob, and an adversary referred to as Eve. Alice aims to transmit a message $\boldsymbol{m}$ to Bob, such that Bob reliably retrieves the message while keeping Eve oblivious about the message. In 1949, Claude Shannon formulated the problem of *unconditional* secrecy in communication systems in one of his seminal papers [7] using information-theoretic measures. The formulation assumes Alice and Bob share a secret key $\boldsymbol{k}$ and Alice encodes the message $\boldsymbol{m}$ using the key $\boldsymbol{k}$ to produce a codeword $\boldsymbol{c}$ such that Bob can retrieve the message using the key, while Eve gains no information, in an information-theoretic sense, about $\boldsymbol{m}$ after observing $\boldsymbol{c}$.

Shannon's work opened the door for a wide range of research directions examining unconditional secrecy using information-theoretic measures, a field that is now referred to as information-theoretic security. For example, in 1975, Wyner [8] proposed taking advantage of the inherent noise in communication channels to ensure secrecy rather than relying on a key. The setup, known as the wiretap channel setup, assumes the channel between Alice and Eve is a degraded version, i.e., noisier version, of the channel between Alice and Bob. The wiretap setup has spurred many works in designing coding schemes for different channels [9, 10, 11]. Another direction of information-theoretic security is concerned with the generation of common randomness. In 1993, Maurer [12], followed by Ahlswede and Csiszr [13] soon after, proposed a setup for generating shared secret random bit sequences between Alice and Bob using a common source of randomness and public discussion while ensuring Eve's knowledge about the generated sequence is negligible. This setup has seen a lot of research efforts and is the basis of many secret key generation protocols.

Another approach to data security is modern cryptography where security is ensured as long as Eve's capabilities are bounded, i.e., security *conditional* on Eve's resources. Modern cryptography has grown substantially in the computer era, especially with the advent of public-key cryptography. Modern cryptography protocols for data security can be split into two groups: symmetric-key cryptography and public-key cryptography. Nowadays, symmetric-key cryptography is typically used for encryption, e.g., Advanced Encryption Standard (AES) [14], while public-key cryptography is typically used for key distribution, e.g., DiffieHellman key exchange [15].

1.2.1 Key Generation

Symmetric-key encryption mechanisms require the legitimate parties to share the same key beforehand. The current state of key agreement in the modern cryptography literature either employs key distribution by a central entity beforehand or uses public-key cryptography mechanisms to facilitate the key agreement between devices. Key distribution mechanisms require a central entity to generate keys for each pair of devices in the wireless network and communicate with each device to share its keys with all other devices. For wireless networks with a massive number of devices, this process becomes prohibitively expensive and introduces communication and storage overhead. On the other hand, public-key-based mechanisms provide on-demand key agreement as needed by devices but typically require computationally expensive mathematical operations. An alternative to these mechanisms is using the physical layer to generate secret keys following the common randomness model. Fortunately, each pair of devices operating in a wireless environment has access to a common randomness source.

The characteristics of the wireless channel between any two users can be used as the source of common randomness and therefore can be used to generate secret keys. Specifically, Alice and Bob perform channel measurements to acquire n samples $\boldsymbol{h} = [h_1, h_2, h_3, ..., h_n]$ and $\tilde{\boldsymbol{h}} = [\tilde{h}_1, \tilde{h}_2, \tilde{h}_3, ..., \tilde{h}_n]$, at Alice and Bob, respectively. Alice and Bob, through the exchange of messages through a public channel, convert such samples to random bit sequences that can be used for key generation. The reciprocity property of channel coefficients in wireless channels ensures that the measurements $\boldsymbol{h}$ and $\tilde{\boldsymbol{h}}$ are highly correlated. Moreover, the samples will have inherent randomness if each sample is collected during a different coherence time slot, which makes the channel coefficients change randomly as long as some mobility in the surrounding environment exists. The security of the generated sequences is guaranteed as long as the eavesdropper Eve is located a certain distance away from both Alice and Bob. Specifically, If Eve is located more than half a wavelength away from both Alice and Bob, her observations are statistically uncorrelated with both Alice's and Bob's observations. For example, when devices are operating in the 3 GHz band, the required distance to ensure security is around 5 cm.

The aforementioned protocol is suitable for many scenarios where devices exist in highly dynamic environments. However, it fails in a scenario where no significant mobility is observed in the surrounding environments, e.g., IoT devices operating indoors. More specifically, the coherence time in static environments is long, which causes Alice and Bob to observe highly correlated samples over time, i.e., $h_1 \approx h_2 \approx h_3 \approx ... \approx h_n$. Although the security condition is still satisfied, the amount of generated secret bits is very limited. To resolve this issue, many works in the literature have proposed solutions that utilize multiple-input-multiple-output (MIMO) antennas systems, beamforming, deploying friendly jamming, user-introduced randomness, and others [16, 17, 18, 19, 20]. These works require some complex underlying architectures, e.g., MIMO transceivers, or unconstrained sources of randomness that are expensive to implement, especially for devices with limited resources.

1.2.2 Secure Coding

Data security typically has been abstracted from the physical layer and treated as an independent process at upper layers of communication networks. There have been many efforts to implement security schemes at the physical layer by taking advantage of the properties of communication schemes. For example, utilizing properties of orthogonal frequency-division multiplexing (OFDM) and MIMO to ensure security [21, 22, 23, 19], exploiting the reciprocity of wireless channels to obfuscate the transmitted signals [24], and many others. Other works have also combined ideas from cryptography with physical layer encoders for secrecy [25, 26].

Secure coding is not limited to encryption and decryption. For example, secure network coding is an area of research where the transmitter aims to multicast messages to multiple receivers through the network while maintaining an eavesdropper, who has access to some number of links in the network, oblivious about individual messages [27, 28].

Schemes with relaxed security conditions in specific setups have been presented by assuming some restrictions on the eavesdropper. One form of relaxation is assuming limited capabilities on the eavesdropper such as having access to only a limited number of messages, as in the aforementioned secure network coding, secret sharing, and secure distributed storage. In other words, the security of messages is ensured as long as the eavesdropper only has access to a number of messages that does not exceed some *threshold*. Another form of relaxation assumes the existence of some secret shared knowledge between the legitimate parties such that an eavesdropper observing the transmitted message cannot retrieve any part of the message of size up to some *threshold* without knowledge of the shared secret. This is suitable for scenarios where partial knowledge of the message does not provide meaningful information about the content of the message itself. Specifically, successfully deducing meaningful information about the message requires knowledge of a significant portion of the message if not the entire message. That is suitable for applications where data is scrambled, hashed, or masked prior to encoding the message. For example, randomly assigned identification numbers, indices of elements in a dataset, etc.

Implementing secure coding schemes at the physical layer using existing encoders and decoder provides an efficient alternative to cryptographic schemes by allowing hardware resources to be shared across multiple tasks, which is suitable for devices whose resources are limited.

1.3 Privacy and Learning

The increasing amount of generated data by users and devices has enabled the training of very sophisticated machine learning (ML) models. The training process of a specific ML model utilizes the training dataset to fine-tune the model parameters such that it performs well on the training dataset. Training a model requires an iterative process described by an algorithm $A(\cdot)$ that takes the model parameters $\boldsymbol{\theta}$ and the training dataset $\boldsymbol{D}$ as inputs along with some other hyperparameters. The output of the algorithm is the trained model parameters.

Due to the massive interest in machine learning over the past few years, ML models have been a subject of extensive research examining their privacy and security. Threats to ML models vary based on the setup in question. One class of threats is concerned with degrading the model's performance by attacking the system during the training process, known as adversarial training. Examples of such attacks include poisoning the training dataset and generating adversarial samples depending on the model itself. The other class of threats is concerned with inferring information about the training dataset itself using the trained model. The latter class of threats compromises the privacy of users' data used during training. One attack against the privacy of the training dataset is known as the membership inference attack, where the adversary uses the model as a black box to decide with some confidence whether a sample was used in the training of the model. Another attack is known as the model inversion attack, where an adversary has access to the model parameters and attempts to create feature vectors that are correlated with the ones used in training the model. These attacks can reveal a considerable amount of information about samples, see Figure 1.1 for an example.

Recently, there have been many efforts in several countries around the world to protect users'

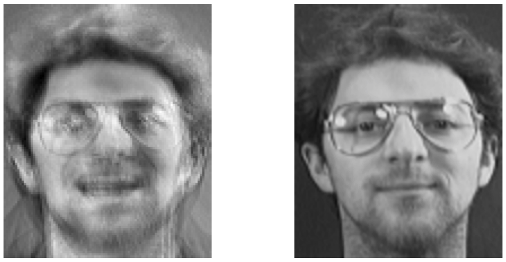


Figure 1.1: An image recovered using a model inversion attack (left) and a training set image of the victim (right). The attacker is given only the persons name and access to a facial recognition system that returns a class confidence score [1].

data. For example, the General Data Protection Regulation has been introduced in the European Union, and California Consumer Privacy Act has been introduced in California. Hence, protecting users' privacy is becoming an essential part of any system using ML models. The level of trust in the central server given by users dictates the appropriate treatment of the privacy requirements. On top of that, if users trust that their privacy is ensured, they are more inclined to share their data, enabling the training of better ML models.

1.3.1 Machine Unlearning

The right to be forgotten in the aforementioned regulations gives a person the right to have personal information removed from internet searches and directories. Deleting data from hard drives only guarantees their removal from storage units. However, ML models trained on these samples store traces of information about them in the model parameters as discussed earlier. As a result, the need to ensure the removal of samples and to satisfy the right to be forgotten requirement has encouraged efforts to explore this problem for ML models. This type of privacy requirement appears in scenarios where ML models are trained on huge datasets on a trusted server. For instance, it shows up when building Machine Learning as a Service (MLaaS) platforms, where models are maintained on the server and not disclosed to other parties. In machine unlearning setups, the adversary is modeled as a person who has access to the ML model as a black box only and cannot copy the model from its owner.

The removal of samples can be guaranteed if the sample is removed from the training dataset and the ML model is retrained from scratch on the updated training dataset. This becomes quite expensive to implement, especially as ML models are trained on large datasets. First coined by [29], *machine unlearning* refers to the process of efficient removal of a sample from the trained ML model. Machine unlearning algorithms should be more efficient and maintain comparable performance as the baseline algorithm that retrains the model from scratch.

Unlearning algorithms are divided into two sets of algorithms. The first set guarantees *perfectly* removal of a sample from the ML model [29, 2], while the second only guarantees the removal of a sample *statistically* [30, 31, 32]. More specifically, perfect unlearning algorithms output a model that is a statistical draw from the distribution of models trained on the updated training dataset with probability 1, while statistical unlearning algorithms output a statistical draw from the distribution of models trained on the updated training dataset with probability at most $1 - \delta$.

A general unlearning algorithm that guarantees perfect unlearning utilizes a similar setup of ensemble learning methods to lower the cost of unlearning of a sample [2]. In this algorithm, the entire training dataset is sharded into small shards with a fixed size, then used to train weak learners followed by an aggregator. Removing a sample requires only retraining the weak learner where such sample appears, and its cost is directly related to the number of samples in the shard. This solution provides a generic framework for many ML architectures that can provide comparable performance to the single learner setup. Nevertheless, as the shard size becomes smaller, the unlearning cost is lowered but some degradation in performance can occur. Due to satisfying the perfect unlearning condition, solutions to address the degradation in performance in such a setup are of interest.

1.3.2 Privacy-Preserving Federated Learning

Training an ML model requires clients to upload their datasets to a central server, where training takes place. Resource-constrained devices generate huge amounts of data every day and uploading such data to a central server causes heavy communication overhead in addition to being a privacy risk, making it costly to train models. Federated learning (FL) was introduced in 2016 by [33], where a framework for clients to collaborative train a model on a central server was presented. In each round of FL, the server sends the model parameters to available clients, who compute their local gradients on their datasets and upload them back to the server. The server only collects the gradients and aggregates them using a specific rule to produce the new global model parameters. FL setups operate under different assumptions than other learning setups. For example, in FL there are some assumptions by default on the algorithm, such as partial client participation, non-independent and identically distributed datasets at each client, the possible existence of malicious clients, heterogeneous device capabilities, and others. Although the FL framework ensures the data remains on-device, variations of the aforementioned threats of data privacy are still applicable and protections against them are desired.

Differential privacy (DP) has been considered the gold standard for mathematical privacy guarantees. It provides quantitative guarantees of the amount of information leaked to an adversary observing the output of a differentially private mechanism. More specifically, DP ensures that the ratio of the distributions of the outputs of the mechanism on two adjacent datasets differing in only one entry is bounded by a constant. Many works in the literature have modified versions of ML algorithms, such as stochastic gradient descent (SGD), that utilize differential privacy [34]. These privacy-preserving algorithms limit the amount of leaked information about samples in the training dataset to the adversary. Typically, utilizing DP causes the performance of the ML model to drop in a relative manner to the amount of privacy guaranteed by the algorithm, i.e., more privacy typically causes more performance degradation.

Variations of privacy-preserving FL frameworks using DP have been studied in the literature. Due to the nature of the FL framework, there are two different types of algorithms depending on whether the clients trust the central server. More specifically, if the clients do not trust the server, then each client employs DP during the local update prior to sending the gradients to the central serve [35, 36, 37]. On the other hand, if the clients employ relaxed constraints against the central server, then the server employs DP during the global model update step to satisfy the privacy requirements [38, 39, 40, 41]. The former is referred to as sample-level (local) DP, while the latter is referred to as client-level (central) DP.

Considering central differential privacy, the server needs to enforce the most strict privacy requirement, dictated by any client, on all clients to meet such privacy guarantees in the baseline private federated learning. As a result, additional, and possibly useful, information sent by clients who do not require such level of privacy gets lost in this process needlessly.

1.4 Contributions

This dissertation is largely focused on security and privacy solutions for resource-constrained devices. In the first part of the dissertation, we aim to tackle data security for such devices operating in wireless environments. The second part is concerned with data privacy in learning tasks. Next, we provide descriptions of each chapter and the contributions of the work in this dissertation.

Chapter 2 focuses on the problem of secret key generation in static environments. Two setups are considered for key generation: secret key generation over a direct link, and secret key generation through an untrusted relay node. To address the issue of low-rate key generation, we propose low-complexity protocols for generating secret keys in the two scenarios by inducing locally-generated randomness at the legitimate parties. We explicitly describe the entire process of generating the key from acquiring samples until converting them into matched secret bit sequences. We analyze the security and reliability of the proposed protocols by providing upper-bounds on the probability of a successful eavesdropping attack by the eavesdropper Eve and an upper bound of the probability of agreeing on mismatched key bits by the legitimate parties, respectively. We provide simulation results of the performance of the proposed protocols for two channel models: the fading channel model, and a realistic 5G millimeter wave (mmWave) channel model [42, 43].

Chapter 3 formulates the threshold security condition using information-theoretic metrics. A general coding scheme for noiseless channels satisfying threshold security based on linear block codes is described where the threshold security parameter is directly related to the minimum distance of the linear block code. Moreover, a low-complexity threshold-secure code construction based on Reed-Muller (RM) codes is described, along with its decoder. Furthermore, the setup is extended to noisy channels between the legitimate parties, while the eavesdropper observes the codeword noise-free. A low-complexity, robust, and threshold-secure code construction based on RM codes is also described [44, 45].

Chapter 4 explores perfect machine unlearning for regression problems. An ensemble learning setup is used where the data is sharded prior to training the weak learners. Since the cost of unlearning is directly proportional to the size of the shard, we propose using a method where datasets are compressed using random encoders prior to training. We show that the proposed algorithm satisfies the perfect unlearning condition. Additionally, we conduct experiments to show the trade-off between performance and unlearning cost of the proposed algorithm potentially outperforming the baseline uncoded unlearning algorithm. We provide some insights on whether it is expected to observe a better trade-off for the proposed algorithm for a specific dataset based on some of its properties [46].

Chapter 5 presents a variation of privacy-preserving federated learning using differential privacy. We propose a new setup where clients in the federated learning setup desire privacy by default but are given the option to opt out of it. The server in our setup is aware of the privacy choices and can use that information to its advantage in terms of providing better performance of the resulting model. We present the FeO2 learning algorithm, which incurs minimal modifications to the original differentially private federated averaging algorithm and discuss the hyperparameters that surface due to the new privacy setup. To demonstrate the algorithm's viability, we consider the simple task of federated linear regression and show the success of the algorithm over the vanilla federated averaging algorithm. Additionally, simulation results on the various synthetic and realistic federated datasets are presented to show the success of the proposed algorithm compared to the baseline private federated learning algorithm [47].

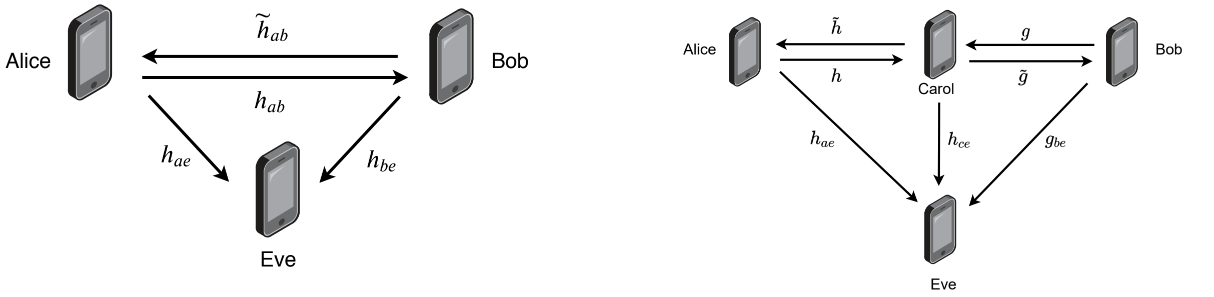
Finally, Chapter 6 concludes the dissertation by providing a summary of the contributions along with discussions on possible directions of future work.

CHAPTER 2

Secret Key Generation

2.1 Introduction

This chapter presents secret key generation protocols for static environments. Secret key generation protocols in the literature require the environment in which devices operate to be dynamic, which limits their applications to a few setups. In this chapter, we present two protocols for secret key generation in static environments. Due to the long channel coherence time, the channel coefficients remain constant over a long period of time; hence, limiting the amount of randomness that can be extracted. To overcome this issue, we propose inducing randomness at the legitimate parties over subcarriers, where Alice and Bob generate random bits, then map them to quadrature amplitude modulation (QAM) symbols, and send them to each other. Using the received signals and the locally generated symbols, Alice and Bob are able to generate correlated samples that can be used to generate symmetric secret keys. Post-processing is performed to generate bit sequences and verify they match at the legitimate parties. In scenarios where direct channels are not available, secret key generation through a relay, referred to as Carol, is also presented. In such case, for networks where devices can be easily hacked, such as IoT networks, it is important to ensure the amount of leaked information to the relay is minimal.



(a) (b)

Figure 2.1: System model for (a) direct key generation, (b) relay-based key generation.

The proposed protocols overcome the limitations of resource-constrained devices operating in static environments by not requiring any complex architectures to be implemented on devices. In addition to the simulations to measure the performance of the protocols, the security evaluation considers a stricter assumption on the uncorrelatedness of the channels between Alice and Bob and Alice and Eve. Bounds on the probability of successful attacks are presented to show the protocols' security. It is worth noting that although the protocol is designed to operate in static channels, it can also operate in dynamic environments given each two-way exchange is done within a coherence time slot.

2.2 Proposed Secret Key Generation Protocols

The setup of the considered secret key generation (SKG) system is shown in Figure 2.1a. The considered wireless channel is assumed to be a fading channel. Suppose that Alice transmits a signal $x_{\text{Alice}}(t)$ to Bob, he receives

$$y_{\text{Bob}}(t) = x_{\text{Alice}}(t) \circledast h_{ab}(t) + n_b(t), \tag{2.2.1}$$

where t denotes the time, $\circledast$ denotes the convolution operator, $h_{ab}(t)$ denotes the circularly-symmetric Gaussian-distributed channel response with mean 0 and variance $\sigma_h^2/2$ in each dimension, and $n_b(t)$ denotes the circularly-symmetric Gaussian-distributed additive noise component with mean 0 and variance $\sigma_n^2/2$ in each dimension. In the case of flat fading channels, the convolution converts to multiplication. The same applies when Bob transmits a signal to Alice through the channel $\widetilde{h}_{ab}$, and the channel coefficients are reciprocal, i.e., $h_{ab} \approx \widetilde{h}_{ab}$. When Alice and Bob utilize OFDM, the transmitted signal can be expressed as a vector, whose j-th element is the j-th symbol, denoted as $\boldsymbol{x}_{\text{Alice}}(t)$. The j-th symbol is transmitted over the j-th subcarrier, the received signal at Bob is expressed as the following vector

$$\boldsymbol{y}_{\text{Bob}}(t) = \boldsymbol{x}_{\text{Alice}}(t) \circ \boldsymbol{h}_{ab}(t) + \boldsymbol{n}_b(t), \tag{2.2.2}$$

where $\circ$ denotes the Hadamard product, i.e., the element-wise product. The relay node, i.e., Carol, employs an amplify-and-forward (AF) function with amplification factor α_g, which multiplies the received signal by α_g then transmits it to the intended receiver. Extending the setup to the relay-based scenario follows straightforwardly.

The proposed secret key generation protocols for both scenarios, i.e., when a direct channel is available and when communication is only available via a relay, can be partitioned into three stages: induced randomness exchange, quantization together with reconciliation, and privacy amplification together with consistency checking. The first stage, i.e., induced randomness exchange, is done differently in the two considered scenarios, while the remaining stages are similar. For better readability, we drop the time t and replace it with round index i, since the channel is static over any two-way exchange. Notations for various vectors in the protocols are summarized in Table

Table 2.1: Notation Summary for the i-th SKG Session

Symbol	Description
$\boldsymbol{p}_s$	Known channel probing vector
$\boldsymbol{r}_{a,i}$	Alice's local randomness
$\boldsymbol{r}_{b,i}$	Bob's local randomness
$\boldsymbol{h}_{ab,i}$	Channel coefficients from Alice to Bob
$\widetilde{\boldsymbol{h}}_{ab,i}$	Channel coefficients from Bob to Alice
$\boldsymbol{h}_i$	Channel coefficients from Alice to the relay
$\widetilde{\boldsymbol{h}}_i$	Channel coefficients from the relay to Alice
$\boldsymbol{g}_i$	Channel coefficients from Bob to the relay
$\widetilde{\boldsymbol{g}}_i$	Channel coefficients from the relay to Bob
$\boldsymbol{h}_{ae,i}$	Channel coefficients from Alice to Eve
$\boldsymbol{h}_{be,i}$	Channel coefficients from Bob to Eve
$\boldsymbol{r}_{ab,i}$	Alice's samples used for quantization
$\widetilde{\boldsymbol{r}}_{ab,i}$	Bob's samples used for quantization
$\boldsymbol{q}_{a,i}$	Alice's quantized version of $\boldsymbol{r}_{ab,i}$
$\boldsymbol{q}_{b,i}$	Bob's quantized version of $\widetilde{\boldsymbol{r}}_{ab,i}$
$\boldsymbol{k}_{ab,i}$	Alice's key bits
$\widetilde{\boldsymbol{k}}_{ab,i}$	Bob's key bits
$\boldsymbol{ch}_{ab,i}$	Alice's check sequence bits
$\widetilde{\boldsymbol{ch}}_{ab,i}$	Bob's check sequence bits

2.1. Next, we expand on each stage of a single session of the protocol.

2.2.1 Induced Randomness Exchange

In this stage, Alice and Bob aim at creating highly correlated yet random observations by exchanging signals. Assume that Alice and Bob use N_c OFDM subcarriers in this protocol.

2.2.1.1 Direct Induced Randomness Exchange

Alice and Bob exchange randomly generated symbols with each other. In the i-th session, Alice chooses a vector $\boldsymbol{r}_{a,i}$ of length N_c and Bob also chooses a vector $\boldsymbol{r}_{b,i}$ of length N_c. Each element of the vectors $\boldsymbol{r}_{a,i}$ and $\boldsymbol{r}_{b,i}$ is chosen independently and uniformly at random from a set of M symbols in a M-QAM constellation. Then, the symbols are multiplied by a pulse/carrier signal for transmission. The reason behind choosing the symbols from M-QAM constellation is that

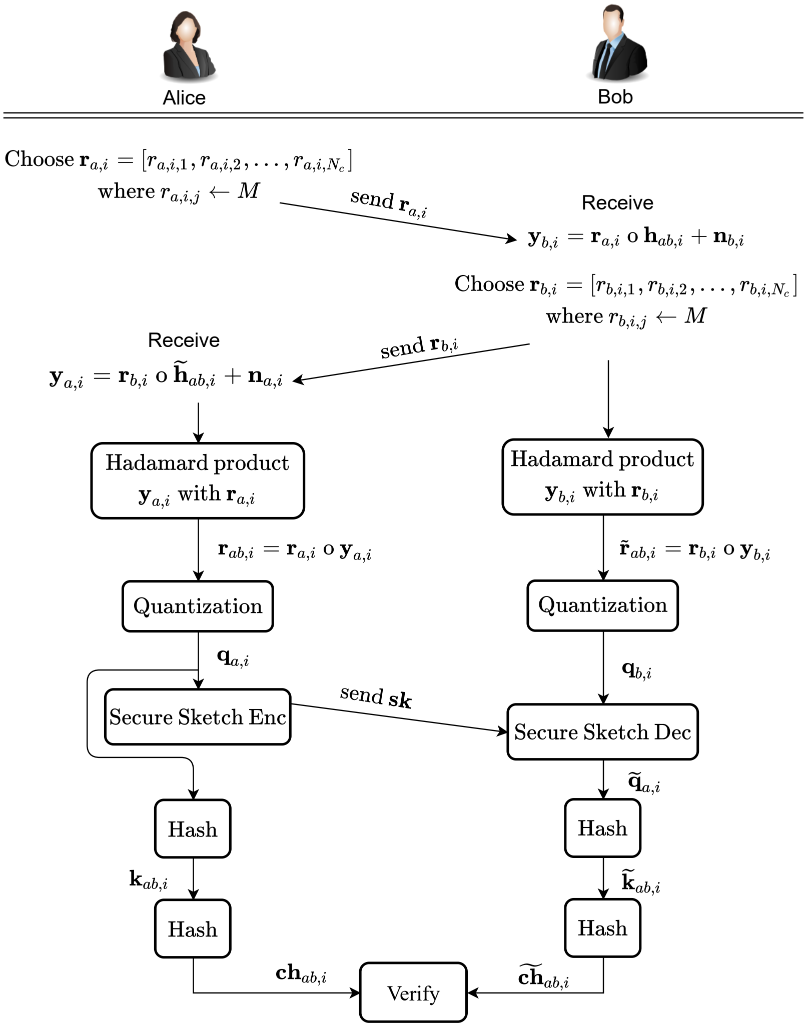
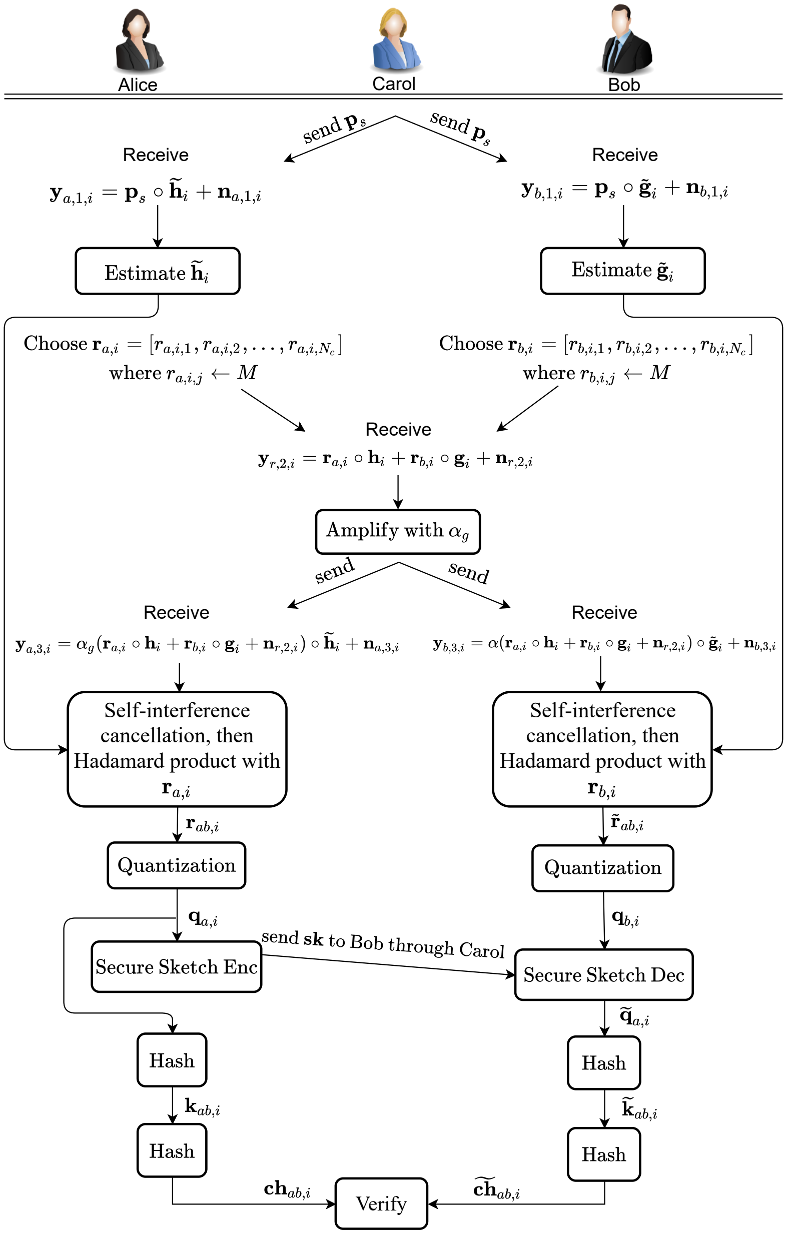

Alice
Bob
Choose $\mathbf{r}_{a,i} = [r_{a,i,1}, r_{a,i,2}, \ldots, r_{a,i,N_c}]$
where $r_{a,i,j} \leftarrow M$
send $\mathbf{r}_{a,i}$
Receive
$\mathbf{y}_{b,i} = \mathbf{r}_{a,i} \circ \mathbf{h}_{ab,i} + \mathbf{n}_{b,i}$
Choose $\mathbf{r}_{b,i} = [r_{b,i,1}, r_{b,i,2}, \ldots, r_{b,i,N_c}]$
where $r_{b,i,j} \leftarrow M$
send $\mathbf{r}_{b,i}$
Receive
$\mathbf{y}_{a,i} = \mathbf{r}_{b,i} \circ \widetilde{\mathbf{h}}_{ab,i} + \mathbf{n}_{a,i}$
Hadamard product $\mathbf{y}_{a,i}$ with $\mathbf{r}_{a,i}$
Hadamard product $\mathbf{y}_{b,i}$ with $\mathbf{r}_{b,i}$
$\mathbf{r}_{ab,i} = \mathbf{r}_{a,i} \circ \mathbf{y}_{a,i}$
$\tilde{\mathbf{r}}_{ab,i} = \mathbf{r}_{b,i} \circ \mathbf{y}_{b,i}$
Quantization
Quantization
$\mathbf{q}_{a,i}$
$\mathbf{q}_{b,i}$
Secure Sketch Enc
send $\mathbf{sk}$
Secure Sketch Dec
$\widetilde{\mathbf{q}}_{a,i}$
Hash
Hash
$\mathbf{k}_{ab,i}$
$\widetilde{\mathbf{k}}_{ab,i}$
Hash
Hash
$\mathbf{ch}_{ab,i}$
Verify
$\widetilde{\mathbf{ch}}_{ab,i}$

Figure 2.2: Direct secret key generation protocol overview of a single session.

the hardware for transmitting and receiving QAM symbols is readily available in many wireless devices. After the exchange of random symbols, Alice and Bob multiply what they sent with what they received. This results in random sequences $\boldsymbol{r}_{ab,i}$ and $\widetilde{\boldsymbol{r}}_{ab,i}$ available at Alice and Bob, respectively, as follows:

$$\boldsymbol{r}_{ab,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{r}_{b,i} \circ \widetilde{\boldsymbol{h}}_{ab,i} + \boldsymbol{r}_{a,i} \circ \boldsymbol{n}_{a,i}, \tag{2.2.3}$$

$$\widetilde{\boldsymbol{r}}_{ab,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{r}_{b,i} \circ \boldsymbol{h}_{ab,i} + \boldsymbol{r}_{b,i} \circ \boldsymbol{n}_{b,i}. \tag{2.2.4}$$



Alice
Carol
Bob
send $\mathbf{p}_s$
send $\mathbf{p}_s$
Receive
$\mathbf{y}_{a,1,i} = \mathbf{p}_s \circ \widetilde{\mathbf{h}}_i + \mathbf{n}_{a,1,i}$
Receive
$\mathbf{y}_{b,1,i} = \mathbf{p}_s \circ \tilde{\mathbf{g}}_i + \mathbf{n}_{b,1,i}$
Estimate $\widetilde{\mathbf{h}}_i$
Estimate $\tilde{\mathbf{g}}_i$
Choose $\mathbf{r}_{a,i} = [r_{a,i,1}, r_{a,i,2}, \ldots, r_{a,i,N_c}]$
where $r_{a,i,j} \leftarrow M$
Choose $\mathbf{r}_{b,i} = [r_{b,i,1}, r_{b,i,2}, \ldots, r_{b,i,N_c}]$
where $r_{b,i,j} \leftarrow M$
Receive
$\mathbf{y}_{r,2,i} = \mathbf{r}_{a,i} \circ \mathbf{h}_i + \mathbf{r}_{b,i} \circ \mathbf{g}_i + \mathbf{n}_{r,2,i}$
Amplify with α_g
send
send
Receive
$\mathbf{y}_{a,3,i} = \alpha_g(\mathbf{r}_{a,i} \circ \mathbf{h}_i + \mathbf{r}_{b,i} \circ \mathbf{g}_i + \mathbf{n}_{r,2,i}) \circ \widetilde{\mathbf{h}}_i + \mathbf{n}_{a,3,i}$
Receive
$\mathbf{y}_{b,3,i} = \alpha(\mathbf{r}_{a,i} \circ \mathbf{h}_i + \mathbf{r}_{b,i} \circ \mathbf{g}_i + \mathbf{n}_{r,2,i}) \circ \tilde{\mathbf{g}}_i + \mathbf{n}_{b,3,i}$
Self-interference cancellation, then Hadamard product with $\mathbf{r}_{a,i}$
Self-interference cancellation, then Hadamard product with $\mathbf{r}_{b,i}$
$\mathbf{r}_{ab,i}$
$\tilde{\mathbf{r}}_{ab,i}$
Quantization
Quantization
$\mathbf{q}_{a,i}$
$\mathbf{q}_{b,i}$
Secure Sketch Enc
send $\mathbf{sk}$ to Bob through Carol
Secure Sketch Dec
$\widetilde{\mathbf{q}}_{a,i}$
Hash
Hash
$\mathbf{k}_{ab,i}$
$\widetilde{\mathbf{k}}_{ab,i}$
Hash
Hash
$\mathbf{ch}_{ab,i}$
Verify
$\widetilde{\mathbf{ch}}_{ab,i}$

Figure 2.3: Relay-based secret key generation protocol overview of a single session.

These two vectors are randomized and highly correlated, as will be shown, which makes them suitable for extracting shared secret keys between Alice and Bob.

2.2.1.2 Relay-Based Induced Randomness Exchange

The relay first transmits a known probing vector $\boldsymbol{p}_s$ to Alice and Bob, who receive $\boldsymbol{y}_{a,1,i}$ and $\boldsymbol{y}_{b,1,i}$, respectively, specified as follows:

$$\boldsymbol{y}_{a,1,i} = \boldsymbol{p}_s \circ \widetilde{\boldsymbol{h}}_i + \boldsymbol{n}_{a,1,i}, \tag{2.2.5}$$

$$\boldsymbol{y}_{b,1,i} = \boldsymbol{p}_s \circ \widetilde{\boldsymbol{g}}_i + \boldsymbol{n}_{b,1,i}. \tag{2.2.6}$$

Alice and Bob then estimate the channels between themselves and the relay, i.e., $\widetilde{\boldsymbol{h}}_i$ and $\widetilde{\boldsymbol{g}}_i$, respectively, using their observations. Their estimates are denoted by $\widehat{\boldsymbol{h}}_i$ and $\widehat{\boldsymbol{g}}_i$ with estimation errors defined as $\boldsymbol{n}_{h,i} = (\boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i - \widehat{\boldsymbol{h}}_i^{\circ 2})$ and $\boldsymbol{n}_{g,i} = (\boldsymbol{g}_i \circ \widetilde{\boldsymbol{g}}_i - \widehat{\boldsymbol{g}}_i^{\circ 2})$, respectively, where $(.)^{\circ 2}$ denotes the element-wise square operation. Alice and Bob utilize their respective channel estimates together with their respective local randomness to eliminate the self-interference terms and to generate the correlated samples, to be described next.

Alice and Bob generate, independently and uniformly at random, vectors of length N_c consisting of M-QAM symbols. Let $\boldsymbol{r}_{a,i}$ and $\boldsymbol{r}_{b,i}$ denote Alice's and Bob's vectors, respectively. They use the probing vector $\boldsymbol{p}_s$ also for synchronization and, simultaneously, transmit their vectors to the relay in such a way that the received SNRs at the relay with respect to the received sequences from Alice and Bob are the same, and equal to a predetermined value. The relay receives

$$\boldsymbol{y}_{r,2,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{h}_i + \boldsymbol{r}_{b,i} \circ \boldsymbol{g}_i + \boldsymbol{n}_{r,2,i}. \tag{2.2.7}$$

Then, it amplifies $\boldsymbol{y}_{r,2,i}$ with an amplification factor α_g, which is chosen to meet a specific SNR at Alice and Bob, and forwards the amplified signal to Alice and Bob who receive $\boldsymbol{y}_{a,3,i}$ and $\boldsymbol{y}_{b,3,i}$, respectively, as follows:

$$\boldsymbol{y}_{a,3,i} = \alpha_g(\boldsymbol{r}_{a,i} \circ \boldsymbol{h}_i + \boldsymbol{r}_{b,i} \circ \boldsymbol{g}_i + \boldsymbol{n}_{r,2,i}) \circ \widetilde{\boldsymbol{h}}_i + \boldsymbol{n}_{a,3,i}, \tag{2.2.8}$$

$$\boldsymbol{y}_{b,3,i} = \alpha_g(\boldsymbol{r}_{a,i} \circ \boldsymbol{h}_i + \boldsymbol{r}_{b,i} \circ \boldsymbol{g}_i + \boldsymbol{n}_{r,2,i}) \circ \widetilde{\boldsymbol{g}}_i + \boldsymbol{n}_{b,3,i}. \tag{2.2.9}$$

The value of the amplification factor α_g is assumed to be publicly known. Alice and Bob utilize what they receive from the relay together with their locally generated vectors, their channel estimates, and α_g in order to construct highly correlated samples. More specifically, the self-interference terms $\alpha_g\boldsymbol{r}_{a,i} \circ \boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i$ and $\alpha_g\boldsymbol{r}_{b,i} \circ \boldsymbol{g}_i \circ \widetilde{\boldsymbol{g}}_i$ are cancelled at Alice and Bob, respectively, using their local randomness and the channel estimates. The results are normalized by α_g and then multiplied by the local randomness, which results in $\boldsymbol{r}_{ab,i}$ and $\widetilde{\boldsymbol{r}}_{ab,i}$ at Alice and Bob, respectively, as follows:

$$\boldsymbol{r}_{ab,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{r}_{b,i} \circ \boldsymbol{g}_i \circ \widetilde{\boldsymbol{h}}_i + \widehat{\boldsymbol{n}}_{a,3,i}, \tag{2.2.10}$$

$$\widetilde{\boldsymbol{r}}_{ab,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{r}_{b,i} \circ \widetilde{\boldsymbol{g}}_i \circ \boldsymbol{h}_i + \widehat{\boldsymbol{n}}_{b,3,i}, \tag{2.2.11}$$

where

$$\widehat{\boldsymbol{n}}_{a,3,i} = \boldsymbol{r}_{a,i}^{\circ 2} \circ \boldsymbol{n}_{h,i} + \boldsymbol{r}_{a,i} \circ \boldsymbol{n}_{r,2,i} \circ \widetilde{\boldsymbol{h}}_i + \frac{1}{\alpha_g}\boldsymbol{r}_{a,i} \circ \boldsymbol{n}_{a,3,i}, \tag{2.2.12}$$

$$\widehat{\boldsymbol{n}}_{b,3,i} = \boldsymbol{r}_{b,i}^{\circ 2} \circ \boldsymbol{n}_{g,i} + \boldsymbol{r}_{b,i} \circ \boldsymbol{n}_{r,2,i} \circ \widetilde{\boldsymbol{g}}_i + \frac{1}{\alpha_g}\boldsymbol{r}_{b,i} \circ \boldsymbol{n}_{b,3,i}, \tag{2.2.13}$$

are the noise terms. The two vectors $\boldsymbol{r}_{ab,i}$ and $\widetilde{\boldsymbol{r}}_{ab,i}$ observed by Alice and Bob are highly correlated and randomized at each session, which makes them suitable for extracting secret keys.

2.2.2 Quantization and Reconciliation

The complex-valued shared sequences $\boldsymbol{r}_{ab,i}$ and $\widetilde{\boldsymbol{r}}_{ab,i}$ need to be quantized into bit streams. The vectors are sorted into a vector of length $2N_c$ whose elements are the ordered real and imaginary parts of the observations. Then, Alice and Bob find the range of sorted data, computed as the difference between the maximum value and the minimum value of the sorted vector. Then, using the range and the quantization resolution δ_q, they identify $\Delta_q = 2^{\delta_q}$ uniform quantization intervals and assign a Gray-code sequence to each interval. Finally, they map each sample to its quantized bit sequence based on the interval it belongs to. The resulting bit sequences for Alice and Bob are denoted by $\boldsymbol{q}_{a,i}$ and $\boldsymbol{q}_{b,i}$, respectively.

Due to the noise in the observations vectors $\boldsymbol{r}_{ab,i}$ and $\widetilde{\boldsymbol{r}}_{ab,i}$, the quantized bit sequences are expected to have some mismatched bits that need to be corrected. The aim of reconciliation is to mitigate such mismatches between Alice's and Bob's quantized bit sequences. To this end, various methods, such as error-correcting codes, can be used. In our protocols, we use error-correcting code-based secure sketch [48], while picking a convolutional code as the underlying code. The reason to pick convolutional codes is due to the simplicity of the encoding process using shift registers and the decoding process using Viterbi decoders [49]. A formal definition of a general secure sketch scheme is as follows:

Definition 2.1 (secure sketch) *[50] An $(\mathcal{S}, m_1, m_2, d_t)$-secure sketch scheme consists of a sketch function $SS(\cdot)$, and a recovery function $SR(\cdot)$ such that the following hold:*

1. *The sketch function takes an input $r_s \in \mathcal{S}$ and returns a randomized $SS(r_s) \in \{0,1\}^*$.*
2. *The recovery function $SR(\cdot)$ takes $SS(r_s)$ and $\tilde{r_s} \in \mathcal{S}$, and returns r_s with probability one as long as the distance between r_s and $\tilde{r_s}$ is less than or equal a certain threshold d_t.*

3. *For any random variable R_s over $\mathcal{S}$ with min-entropy m_1, an adversary observing $SS(R_s)$ has an average min-entropy of R_s conditioned on $SS(R_s)$ as $\tilde{H}_\infty(R_s|SS(R_s)) \geqslant m_2$.*

Note that the min-entropy function of a random variable X is computed as $H_\infty(X) = -\log_2(\max_x(\mathrm{Pr(X = x)}))$ *and the average min-entropy function of X conditioned on Y is computed as* $\tilde{H}_\infty(X|Y) = -\log_2\big(\mathop{\mathbb{E}}_{y \leftarrow Y}[2^{-H_\infty(X|Y=y)}]\big)$.

Next, we describe a construction known as the code-offset secure sketch [48] using convolutional codes. The encoder is chosen in such a way that the length of its output is equal to the length of $\boldsymbol{q}_{a,i}$. Once the quantized sequences $\boldsymbol{q}_{a,i}$ and $\boldsymbol{q}_{b,i}$ are available, Alice chooses a bit string $\boldsymbol{t}$ uniformly at random and encodes it using the convolutional encoder to get $\mathrm{Enc}(\boldsymbol{t})$, which is of the same length as $\boldsymbol{q}_{a,i}$. Then, she computes

$$\boldsymbol{sk} = \boldsymbol{q}_{a,i} \oplus \mathrm{Enc}(\boldsymbol{t}), \tag{2.2.14}$$

where $\oplus$ is the addition modulo 2, and transmits the resulting sequence over the noiseless public channel, either directly as in the first scenario or through the relay as in the second scenario, to Bob. Then, Bob takes the addition modulo 2 of $\boldsymbol{sk}$ and $\boldsymbol{q}_{b,i}$, feeds it to the Viterbi decoder to get $\widetilde{\boldsymbol{t}}$, and re-encodes $\widetilde{\boldsymbol{t}}$ to get $\mathrm{Enc}(\widetilde{\boldsymbol{t}})$. He computes the final sequence as

$$\begin{aligned} \widetilde{\boldsymbol{q}}_{a,i} &= \boldsymbol{sk} \oplus \mathrm{Enc}(\mathrm{Dec}(\boldsymbol{sk} \oplus \boldsymbol{q}_{b,i})) \\ &= \boldsymbol{sk} \oplus \mathrm{Enc}(\widetilde{\boldsymbol{t}}). \end{aligned} \tag{2.2.15}$$

A binary linear code of length n and dimension m with minimum distance $2d_t + 1$ can be used to build an $(\mathcal{A}^n, m_1, m_1 - (n-m), d_t)$-secure sketch scheme, where $\mathcal{A} = \{0, 1\}$ for binary codes

[50]. The error correction capability of the linear code is related to the underlying rate of the code. This introduces a trade-off between the error correction capability and the security, as higher rates provide better security but can correct fewer errors, and vice versa. Alice and Bob should start with an initial high rate code and then reduce it accordingly if they observe several consecutive unsuccessful attempts of the protocol.

2.2.3 Privacy Amplification and Consistency Checking

Some information about the shared key is leaked to Eve during the exchange of random symbols and the reconciliation stages. To compensate for such leakage, we exploit universal hash functions (UHF). In general, UHFs are desired in such scenarios due to their resilience against collisions.

Definition 2.2 (universal hash function) *[51] A family of hash functions H_f that maps a set of inputs U, e.g., binary vectors of length n, to a value in the hash table of size t_h is called universal if for any two inputs $x, y \in U$ with $x \neq y$, we have*

$$\Pr_{h \leftarrow H_f}(h(x) = h(y) | x \neq y) \leqslant \frac{1}{t_h}. \tag{2.2.16}$$

Given that h should be chosen randomly from H_f, we need to ensure that Alice and Bob agree on the same h. We propose a method that guarantees the same choice of h at Alice and Bob if inputs to the UHF are consistent. Suppose we have a random binary sequences $\boldsymbol{q}_i$ of length n_h (This is $\boldsymbol{q}_{a,i}$ for Alice and $\widetilde{\boldsymbol{q}}_{a,i}$ for Bob). For simplicity, we assume that n_h is an even multiple of some integer $m_h \geqslant 1$. We split $\boldsymbol{q}_i$ into two sequences of equal length $\boldsymbol{q}_i = \boldsymbol{q}_{1,i} || \boldsymbol{q}_{2,i}$ each of length $n_h/2$, which is an integer since n_h is even. Then, $\boldsymbol{q}_{1,i}$ is used to choose h from H_f, and $\boldsymbol{q}_{2,i}$ is used as the input to the hash function h. Next, a well-known construction of UHF is described next, which we use in our protocol [51]. First, the largest prime p with $2^{m_h-1} < p < 2^{m_h}$, i.e., its binary representation consists of m_h bits, is chosen, where m_h is the length of the output bit sequence (such a prime number always exist for $m_h > 1$ by Bertrand's postulate). Then, for $i = 1, 2$, we divide $\boldsymbol{q}_{1,i}$ and $\boldsymbol{q}_{2,i}$ into l_q parts $q_{1,i,l}$ and $q_{2,i,l}$ for $l = 1, 2, \ldots, l_q$, where the length of each part is less than or equal to m_h bits. For ease of notation, let $q_{j,i,l}$ also denote the number with the binary representation $q_{j,i,l}$. Finally, the following summation is computed:

$$h_{\boldsymbol{q}_{1,i}}(\boldsymbol{q}_{2,i}) = \sum_{l=1}^{l_q} q_{1,i,l} q_{2,i,l} \bmod p. \tag{2.2.17}$$

To be able to use this construction, we need to ensure the randomness of $\boldsymbol{q}_i$. In our protocol, $\boldsymbol{r}_{a,i}$ and $\boldsymbol{r}_{b,i}$ are chosen uniformly at random for each key generation session. Hence, the value of $\boldsymbol{q}_i$ is also random. Therefore, the hash function is randomized during each session, which will be verified in the numerical results section. The output of the aforementioned described hash function is the key bit sequences $\boldsymbol{k}_{ab,i}$ for Alice and $\widetilde{\boldsymbol{k}}_{ab,i}$ for Bob, which are matched with high probability after using the reconciliation step.

We also use UHFs to check consistency between keys generated by Alice and Bob, without leaking any information to Eve, as suggested in [52]. Before Alice and Bob are able to use the key sequences for encryption and decryption, they need to verify the consistency of their keys. To this end, Alice and Bob hash their key sequences $\boldsymbol{k}_{ab,i}$ and $\widetilde{\boldsymbol{k}}_{ab,i}$ again similar to the previously described process. The output of this step is their respective check sequences $\boldsymbol{ch}_{ab,i}$ and $\widetilde{\boldsymbol{ch}}_{ab,i}$, which they can use to verify whether or not their keys are consistent. It is worth noting that, in our protocol, the length of the check sequences, $\boldsymbol{ch}_{ab,i}$ and $\widetilde{\boldsymbol{ch}}_{ab,i}$, is half the length of the key.

Theorem 2.2.1 *The probability of accepting a mismatched key as consistent by the described pro-*

tocol with hash table size t_c for the check sequence is upper bounded as follows:

$$\Pr(\boldsymbol{ch}_{ab,i} = \widetilde{\boldsymbol{ch}}_{ab,i} | \boldsymbol{k}_{ab,i} \neq \widetilde{\boldsymbol{k}}_{ab,i}) \leqslant \frac{1}{t_c}. \tag{2.2.18}$$

Proof: Follows directly from the definition of universal hash functions, specified in (2.2.16), where the output hash table size is t_c. ■

2.3 Security Evaluation

We evaluate the security of the protocol next. Unlike the typical assumption in the literature, we evaluate the direct secret key generation rigorously rather than rely on assuming independent channel coefficients. The relay-based protocol is evaluated assuming an honest but curious relay, which is due to the fact that relays, such as in IoT scenarios, can be hacked after the key generation session.

2.3.1 Direct Secret Key Generation

In this scenario, Eve's best strategy is to acquire $\boldsymbol{r}_{a,i}, \boldsymbol{r}_{b,i}$ and $\boldsymbol{h}_{ab,i}$. When Alice and Bob exchange signals, Eve receives

$$\boldsymbol{e}_{1,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{h}_{ae,i} + \boldsymbol{n}_{e_1,i}, \tag{2.3.1}$$

$$\boldsymbol{e}_{2,i} = \boldsymbol{r}_{b,i} \circ \boldsymbol{h}_{be,i} + \boldsymbol{n}_{e_2,i}. \tag{2.3.2}$$

If Eve is able to estimate both $\boldsymbol{r}_{a,i}$ and $\boldsymbol{r}_{b,i}$ from her observations in (2.3.1) and (2.3.2) perfectly, she can only create samples of the following form:

$$\boldsymbol{r}_{e_d,1,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{r}_{b,i} \circ \boldsymbol{h}_{ae,i} + \boldsymbol{n}_{e_3,i}, \tag{2.3.3}$$

$$\boldsymbol{r}_{e_d,2,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{r}_{b,i} \circ \boldsymbol{h}_{be,i} + \boldsymbol{n}_{e_4,i}. \tag{2.3.4}$$

Note that she still needs to know $\boldsymbol{h}_{ab,i}$ at all different subcarriers in order to obtain $\boldsymbol{r}_{ab,i}$ and/or $\widetilde{\boldsymbol{r}}_{ab,i}$, as described in (2.2.3) and (2.2.4). Luckily, this is, almost, not possible for Eve as discussed next.

Generally, the Pearson correlation coefficient ρ of the channel fading coefficients at locations separated by distance d_s is computed as follows [53]:

$$\rho = \big(J_0(k_w d_s)\big)^2, \tag{2.3.5}$$

where $J_0(.)$ is the Bessel function of the first kind, and k_w is the wavenumber. Therefore, if the distance between Alice/Bob and Eve d_s is larger than half of a wavelength, e.g., 5 cm in 3 GHz band or 0.54 cm in 28 GHz band, they will experience almost uncorrelated fading channels. Therefore, the leaked information about the generated secret key to Eve is small and is often assumed to be negligible in the literature. However, it is fundamentally important to quantitatively measure the security level, even if the distance between devices is very small and impractical in some settings. An information-theoretic measure of security is the mutual information between the shared random sequence, from which the secure key will be generated, and what Eve observes. If we assume that the effect of quantization is negligible and also assume that Eve can perfectly recover $\boldsymbol{r}_{a,i}$ and $\boldsymbol{r}_{b,i}$, this mutual information is equal to the mutual information between $\boldsymbol{h}_{ab,i}$ and the pair $(\boldsymbol{h}_{ae,i}, \boldsymbol{h}_{be,i})$.

One can assume that Eve is closer to Bob than Alice and hence, only need to consider the mutual information between $\boldsymbol{h}_{ab,i}$ and $\boldsymbol{h}_{ae,i}$, since it is the dominating term. This can be calculated in each subcarrier as stated in the next lemma.

Lemma 2.3.1 *Let h_{bk} and h_{ek} denote the fading coefficients of Bob's and Eve's channels at the k-th subcarrier. Also, let ρ denote the correlation coefficient between h_{bk} and h_{ek}, specified in (2.3.5). Then, the mutual information between h_{bk} and h_{ek} is given by*

$$I(h_{bk}; h_{ek}) = -\log(1-\rho^2) \text{ bits.} \tag{2.3.6}$$

Proof: We have $h_{bk} = h_{bk,I} + jh_{bk,Q}$, and $h_{ek} = h_{ek,I} + jh_{ek,Q}$. $h_{bk,I}, h_{bk,Q}$ are independent and identically distributed as $\mathcal{N}(0, \sigma_b^2/2)$ and $h_{ek,I}, h_{ek,Q}$ are independent and identically distributed as $\mathcal{N}(0, \sigma_e^2/2)$. The real parts of Bob's and Eve's channel coefficients are correlated with the parameter ρ, and the imaginary parts are also correlated with ρ. Then, we have the following covariance matrices:

$$\Sigma_1 = \begin{bmatrix} \sigma_b^2/2 & 0 \\ 0 & \sigma_b^2/2 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} \sigma_e^2/2 & 0 \\ 0 & \sigma_e^2/2 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} \sigma_b^2/2 & 0 & \frac{\rho\sigma_b\sigma_e}{2} & 0 \\ 0 & \sigma_b^2/2 & 0 & \frac{\rho\sigma_b\sigma_e}{2} \\ \frac{\rho\sigma_b\sigma_e}{2} & 0 & \sigma_e^2/2 & 0 \\ 0 & \frac{\rho\sigma_b\sigma_e}{2} & 0 & \sigma_e^2/2 \end{bmatrix}. \tag{2.3.7}$$

The following series of equalities holds:

$$\begin{aligned}
I(h_{bk}; h_{ek}) &= I(h_{bk,I} + jh_{bk,Q}; h_{ek,I} + jh_{ek,Q}) \\
&\overset{(a)}{=} I(h_{bk,I}, h_{bk,Q}; h_{ek,I}, h_{ek,Q}) \\
&\overset{(b)}{=} H_d(h_{bk,I}, h_{bk,Q}) + H_d(h_{ek,I}, h_{ek,Q}) - H_d(h_{bk,I}, h_{bk,Q}, h_{ek,I}, h_{ek,Q}) \\
&\overset{(c)}{=} \frac{1}{2}\log(\det(2\pi e\Sigma_1)) + \frac{1}{2}\log(\det(2\pi e\Sigma_2)) - \frac{1}{2}\log(\det(2\pi e\Sigma_3)) \\
&= \log(\pi e\sigma_b^2) + \log(\pi e\sigma_e^2) - \log((\pi e\sigma_b\sigma_e)^2(1-\rho^2)) \\
&= -\log(1-\rho^2), \qquad (2.3.8)
\end{aligned}$$

where:

(a) holds due to having a one-to-one mapping;

(b) is the expansion of the mutual information expression in terms of differential entropy;

(c) holds by using the well-known expression that the differential entropy of multivariate Gaussian random variables $\boldsymbol{X}^n = (X_1 , X_2 , ..., X_n)$ with covariance matrix Σ_i is $H_d(\boldsymbol{X}^n) = \frac{1}{2}\log(\det(2\pi e\Sigma_i))$; and the rest are simplification steps. ■

Note that as ρ goes to zero, the mutual information, given by Lemma 2.3.1, also goes to zero. The next question, which also applies to any physical layer security scheme that utilizes information-theoretic measures of security, is how to quantitatively characterize the chances of a successful eavesdropping attack by Eve, i.e., guessing the key, given the leaked information? The latter is often measured in terms of semantic security, which is a classical notion of security in cryptosystems [54]. Direct connections between metrics for the information-theoretic security, based on the mutual information, and cryptographic measures of security, including semantic security, are provided in [55]. We use these connections to arrive at the following theorem which characterizes the security of the proposed protocol from the aforementioned perspective:

Theorem 2.3.2 *Let N_c denote the number of subcarriers used in the proposed protocol and δ_q denote the quantization resolution. Then, the probability of a successful eavesdropping attack by Eve is upper bounded as follows:*

$$\Pr(\text{Successful attack}) < \left(2^{-2\delta_q} + \sqrt{2I(h_b; h_e)}\,\right)^{N_c} + 2^{-\delta_q N_c}, \tag{2.3.9}$$

where h_b and h_e denote the fading coefficients of Bob's and Eve's channels at a subcarrier.

Proof: [55, Theorem 5] relates the mutual information between Bob's and Eve's observations to the increase in the probability of a successful eavesdropping attack by Eve given her observations. More specifically, the increase in the latter probability is quantified in terms of the mutual information between Bob's and Eve's observations [55, Theorem 5]. Note that the probability that Eve successfully guesses the bits, with no observations, at a single subcarrier is $2^{-2\delta_q}$. In addition to that, by [55, Theorem 5], the probability that Eve can guess the shared random bits in a single subcarrier, given her observations in this subcarrier, is increased by at most $\sqrt{2I(h_b; h_e)}$ compared to the case where she does not have any observation. Therefore, Eve's probability of successfully guessing these quantized key bits is upper bounded by $2^{-2\delta_q} + \sqrt{2I(h_b; h_e)}$. The probability that Eve can recover the shared randomness over all subcarriers is then given by $\left(2^{-2\delta_q} + \sqrt{2I(h_b; h_e)}\right)^{N_c}$. Note that $I(h_b; h_e)$ is the same across all the subcarriers and is actually computed in terms of ρ in Lemma 2.3.1. If Eve cannot recover all the shared randomness, the probability that she can guess the secret key correctly, by the property of hash functions in the privacy amplification part of our protocol, is at most $2^{-\delta_q N_c}$, when using a key sequence of half the quantized bit sequence length. Utilizing these together with the union bound completes the proof.

■

Numerical values of the probability of successful attack can be computed using Theorem 2.3.2 together with Lemma 2.3.1. For instance, suppose that the distance between Eve and Bob is at least half of a wavelength, and is less than the distance between Eve and Alice. Then, the correlation coefficient ρ is at most 0.09 and by Lemma 2.3.1 the resulting mutual information $I(h_b; h_e)$ is at most 0.01 bits at any of the subcarriers. Suppose that $N_c = 16$ and $\delta_q = 2$, which are also used in the numerical results provided in the next section. Then, by Theorem 2.3.2, the probability of a successful attack by Eve given such parameters is at most $2^{-37} + 2^{-32} < 2^{-31}$.

2.3.2 Relay-based Secret Key Generation

In this scenario, Eve tries to use her observations and the messages transmitted over the public channel to guess $\boldsymbol{r}_{ab,i}$ and/or $\widetilde{\boldsymbol{r}}_{ab,i}$, as described in (2.2.10) and (2.2.11). Her best strategy is to find $\boldsymbol{r}_{a,i}$, $\boldsymbol{r}_{b,i}$, $\boldsymbol{g}_i$ and $\boldsymbol{h}_i$. When Alice and Bob transmit their induced randomness, Eve receives

$$\boldsymbol{r}_{e,1,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{h}_{ae,i} + \boldsymbol{r}_{b,i} \circ \boldsymbol{g}_{be,i} + \boldsymbol{n}_{e5,i}. \tag{2.3.10}$$

However, when Carol, the relay, amplifies and forwards the signal from Alice and Bob, Eve receives

$$\boldsymbol{e}_{3,i} = \alpha_g(\boldsymbol{r}_{a,i} \circ \boldsymbol{h}_i + \boldsymbol{r}_{b,i} \circ \boldsymbol{g}_i + \boldsymbol{n}_{r,2,i}) \circ \boldsymbol{h}_{ce,i} + \boldsymbol{n}_{e6,i}. \tag{2.3.11}$$

Since Eve can estimate the channel coefficients $\boldsymbol{h}_{ce,i}$ from the relay's transmission when it transmits the known probing vector and, also, she knows the value of α_g from the messages over the public channel, she can successfully estimate

$$\boldsymbol{r}_{e,2,i} = \boldsymbol{r}_{a,i} \circ \boldsymbol{h}_i + \boldsymbol{r}_{b,i} \circ \boldsymbol{g}_i + \boldsymbol{n}_{r,2,i}. \tag{2.3.12}$$

In a worst-case scenario from the legitimate parties' perspective, Eve has as much information as the relay has, in addition to her own observations. Note that this coincides with the problem of securing the shared key against the untrusted relay Carol when Eve is at Carol's location. In the remainder of this section, we analyze the probability of a successful eavesdropping attack assuming that the eavesdropper Eve has all the information available to Carol, in addition to her own observations.

Note that the computations involving the spatial correlation parameter of the wireless channels do not help in ensuring security in this scenario as they do in the first scenario with a direct communication channel. Also, the mutual information between $\boldsymbol{r}_{ab,i}$, as described in (2.2.10), and the pair $(\boldsymbol{r}_{e,1,i}, \boldsymbol{r}_{e,2,i})$, as described in (2.3.10) and (2.3.12), respectively, is expected not to be very small as it was in the first scenario. For instance, if this mutual information is greater than 0.5, then using [55, Theorem 5], same as in the proof of Theorem 2.3.2, does not yield a non-trivial upper bound on the probability of a successful eavesdropping attack. Hence, instead of utilizing semantic security, we need to use an alternative approach to relate $I(\boldsymbol{r}_{ab,i}; \boldsymbol{r}_{e,1,i}, \boldsymbol{r}_{e,2,i})$ to the probability of a successful eavesdropping attack. To this end, we use Fano's inequality to bound the probability of successful estimation of the quantized bits $\boldsymbol{q}_{a,i}$ by the eavesdropper in terms of the conditional entropy of the quantized bits $\boldsymbol{q}_{a,i}$ given the eavesdropper's observations $(\boldsymbol{r}_{e,1,i}, \boldsymbol{r}_{e,2,i})$. Note that the latter can be bounded in terms of $I(\boldsymbol{r}_{ab,i};\boldsymbol{r}_{e,1,i},\boldsymbol{r}_{e,2,i})$. The details of this analysis are given next in the proof of Theorem 2.3.3.

To simplify the expressions in the next theorem, let us consider an arbitrary subcarrier and denote the corresponding entries of the vectors $\boldsymbol{r}_{ab,i}$, $\boldsymbol{r}_{e,1,i}$, $\boldsymbol{r}_{e,2,i}$, and $\boldsymbol{q}_{a,i}$ as r_{ab}, $r_{e,1}$, $r_{e,2}$, and q_a, respectively. Note that the result of Theorem 2.3.3 does not depend on the choice of the subcarrier.

Theorem 2.3.3 *Let N_c denote the number of subcarriers used in the proposed protocol and δ_q denote the quantization resolution. Then, the probability of a successful eavesdropping attack by Eve is upper bounded as follows:*

$$\Pr(\text{Successful attack}) < \Big(1 - \frac{H(q_a) - I_{ab,e} - 1}{\log_2(|\mathcal{Q}_A|)}\Big)^{N_c} + 2^{-\delta_q N_c}, \tag{2.3.13}$$

where $I_{ab,e}$ denotes $I(r_{ab};r_{e,1},r_{e,2})$, $\mathcal{Q}_A$ denotes the support of q_a, and $|\mathcal{Q}_A|$ denotes its cardinality.

Proof: Let C denote the event of correct estimation of q_a and E denote the event of erroneous estimation of q_a by the eavesdropper. Then, we have the following

$$\begin{aligned} \Pr(\mathrm{C}) &= 1 - \Pr(\mathrm{E}) && (2.3.14)\\ &\overset{\text{(a)}}{\leqslant} 1 - \frac{H(q_a|r_{e,1},r_{e,2}) - 1}{\log_2(|\mathcal{Q}_A|)} && (2.3.15)\\ &\overset{\text{(b)}}{=} 1 - \frac{H(q_a) - I(q_a;r_{e,1},r_{e,2}) - 1}{\log_2(|\mathcal{Q}_A|)} && (2.3.16)\\ &\overset{\text{(c)}}{\leqslant} 1 - \frac{H(q_a) - I(r_{ab};r_{e,1},r_{e,2}) - 1}{\log_2(|\mathcal{Q}_A|)} && (2.3.17)\\ &\overset{\text{(d)}}{=} 1 - \frac{H(q_a) - I_{ab,e} - 1}{\log_2(|\mathcal{Q}_A|)}, && (2.3.18) \end{aligned}$$

where:

(a) holds by Fano's inequality [56];

(b) is the expansion of conditional entropy;

(c) follows from the data processing inequality because q_a is a deterministic function of r_{ab} and hence, $(r_{e,1}, r_{e,2})$, r_{ab}, and q_a form a Markov chain;

(d) is a change of the notation of $I(r_{ab}; r_{e,1}, r_{e,2})$ to $I_{ab,e}$.

Note that the probability of correctly estimating every bit of $\boldsymbol{q}_a$, denoted by $\Pr(\mathrm{C}_{\mathrm{N_c}})$, is equal to the probability of correctly estimating q_a over all the N_c subcarriers, since the computation of mutual information is the same over all subcarriers. Hence, by using the independence of such events across the N_c subcarriers, we have

$$\Pr(\mathrm{C}_{\mathrm{N_c}}) \leqslant \Big(1 - \frac{H(q_a) - I_{ab,e} - 1}{\log_2(|\mathcal{Q}_A|)}\Big)^{N_c}. \tag{2.3.19}$$

If Eve cannot recover all the shared randomness bits in a single session, the probability that she correctly guesses the secret key, by the property of hash functions in the privacy amplification part of our protocol, is at most $2^{-\delta_q N_c}$. This, together with (2.3.19), and using the union bound complete the proof. ■

Next, we illustrate how Theorem 2.3.3 can be used in a numerical setup to upper bound the probability of a successful eavesdropping attack by Eve. Suppose that Alice and Bob use 64-QAM constellation points to transmit their induced randomness, the received SNR is 23 dB at Alice and Bob in (2.2.8), the quantization parameter δ_q is 2, and the number of subcarriers N_c is 16. Eve is located close to Carol, but at least half a wavelength away from her. Given these parameters the mutual information $I_{ab,e} = I(r_{ab}; r_{e,1}, w_{e,2})$ is numerically estimated as $I_{ab,e} \approx 1.39$ bits, and the entropy of the generated key bits is numerically estimated as $H(q_a) \approx 3.86$ bits. Then, by Theorem 2.3.3, the probability of successful eavesdropping attack is upper bounded, approximately, by $2^{-10.57}$.

2.4 Numerical Results

This section presents simulation results of the proposed protocols using typical key generation metrics. These metrics are briefly described next:

1. Bit Generation Rate (BGR): This measures the number of bits per packet in the quantized sequences generated by Alice and Bob, denoted by $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$, respectively.

2. Bit Mismatch Rate (BMR): This measures the ratio of the number of bits that are mismatched between $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$. This quantity can be also measured at Eve's side. Note that the BMR at Eve should be higher than the BMR measured between Alice and Bob; otherwise, no secret key can be generated.

3. Bit Error Rate (BER): This measures the ratio of the number of bits that do not match in the final key generated by Alice and Bob as the output of the protocol. This quantity can be also measured at Eve's side, which, ideally, should be close to 50%.

4. Randomness: This indicates whether the final key bit sequence generated by the protocol, denoted by $\boldsymbol{k}_{ab}$, is indistinguishable from a random binary bit sequence. This is often tested using the NIST statistical test suite [57].

In addition to the aforementioned metrics, we introduce a new parameter, referred to as *randomness efficiency*, to measure the length of the shared sequence normalized by the total amount of randomness available to Alice and Bob. Let R_Q denote the total number of shared random bits after quantization. The randomness efficiency, denoted by E_R, is defined as

$$E_R \overset{\text{def}}{=} \frac{R_Q}{H(R_a) + H(R_b)}, \tag{2.4.1}$$

where $H(R_a)$ and $H(R_b)$ are the entropy of Alice's and Bob's sources of randomness, respectively.

Next, the setups for the simulation results are provided, then we present the numerical results for each setup.

2.4.1 Setup

2.4.1.1 Direct Secret Key Generation

In this scenario, it is assumed that Alice and Bob communicate over a direct and reciprocal wireless channel. The constellation size for each subcarrier is $M = 16$, i.e., the set of 16-QAM symbols is used as the set from which local randomness is chosen and transmitted by Alice and Bob. Also, $N_c = 16$ OFDM subcarriers are assumed to be available in the channel between Alice and Bob. The quantization is done with $\delta_q = 2$, i.e., the real and imaginary parts of the received symbol in each subcarrier are quantized into one of the four possibilities as discussed in Section 2.2.2. Finally, the remaining steps including secure sketch using a convolutional code with rate $1/2$, hashing, and consistency checking are performed as discussed in Section 2.2.

2.4.1.2 NYUSIM-Based Secret Key Generation

In this scenario, it is assumed that Alice and Bob have a direct reciprocal wireless channel where the coefficients are generated by the NYUSIM Channel Simulator [58]. They operate in a nonline-of-sight (NLOS) urban micro-cellular environment at $20°$C, the operating frequency is 28 GHz, and the distance between Alice and Bob and Alice and Eve is 10 meters. The path contains 1 meter of foliage, and there is an outdoor-to-indoor low loss. Channel coefficients between Alice and Bob and channel coefficients between Alice and Eve are generated by the NYUSIM Channel Simulator over $N_c = 16$ subcarriers. Alice and Bob choose their induced randomness from the set of 16-QAM symbols, and the quantization is done with $\delta_q = 2$. The remaining steps follow as in the first scenario.

2.4.1.3 Relay-Based Secret Key Generation

In this scenario, it is assumed that Alice and Bob have direct and reciprocal wireless channels with the relay, which can be perfectly estimated. Also, a scenario is considered for eavesdropping, as discussed in Section 2.3.2, where Eve uses the relay's observations. Alice and Bob choose their induced randomness from the set of 64-QAM symbols, and set their power levels in such a way that the average received SNRs at the relay from both Alice and Bob are equal. Then, the amplification vector α_g is chosen such that the average SNR, which is the one considered in the results, of Alice and Bob's correlated observations, (2.2.10) and (2.2.11), respectively, is the same. The remaining parameters and steps are similar to the previous scenarios.

2.4.2 Results

2.4.2.1 Bit Generation Rate

For all the setups described above, Alice and Bob exchange their induced randomness over $N_c = 16$ subcarriers, with quantization resolution $\delta_q = 2$ for the real and imaginary parts separately. Note that $16 \times 2 \times 2 = 64$ bits are generated by Alice and Bob during each session of the protocol. Hence, the bit generation rate (BGR) is 64 bits/packet. The length of the final secret key is $64/2 = 32$ bits. In order to increase Eve's bit error rate, we assume that four blocks of keys, generated during four separate successful sessions, are added together modulo 2 to obtain one final key of length 32 for every four sessions. Such BGR is considered high compared to protocols designed for static channels setups, which have BGR of $\frac{1}{4}$ to $\frac{1}{2}$ bits/packet as in [20], or 8 bits/packet as in [59], and it is comparable with protocols designed for dynamic environments, such as [60] whose BGR is $60 - 90$ bits/packet.

2.4.2.2 Bit Mismatch Rate and Bit Error Rate

The bit mismatch rate (BMR) and bit error rate (BER) between Alice and Bob, and Alice and Eve are shown in Figure 2.4 and Figure 2.5, respectively, for the three described setups. For the bit mismatch rate, in the direct and NYUSIM-based SKG setups we compare Alice's and Bob's quantized sequences of (2.2.3) and (2.2.4), respectively, and Alice's and Eve's quantized sequences of (2.2.3) and (2.3.3), respectively. Also, for the relay-based SKG setup, we compare Alice's and Bob's quantized sequences of (2.2.10) and (2.2.11), respectively, and Alice's and Eve's quantized sequences of (2.2.10) and (2.3.12), respectively. It is worth noting that as the average SNR increases in the NYUSIM-based SKG setup, Eve's BMR decreases but the rate of decrease slows down. It can be observed that an increase of around 3 dB of the average SNR is required in the relay-based SKG setup to achieve a BMR similar to the first two setups. In comparison with other protocols for static environments at 20 dB, they have BMR of around 1% as in [20], 4% as in [59], 4% and 13% for the direct and relay-based setups as in [61].

As for the BER, we compare Alice's and Bob's final key sequences and Alice's and Eve's final

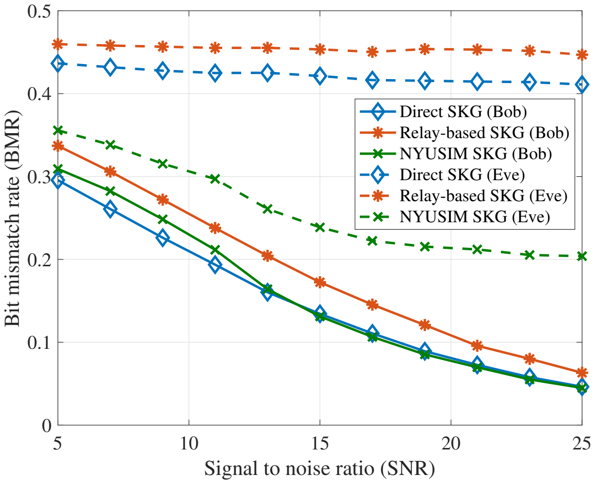


Figure 2.4: The bit mismatch rate (BMR) between Alice's sequence and Bob's and Eve's sequences versus SNR.

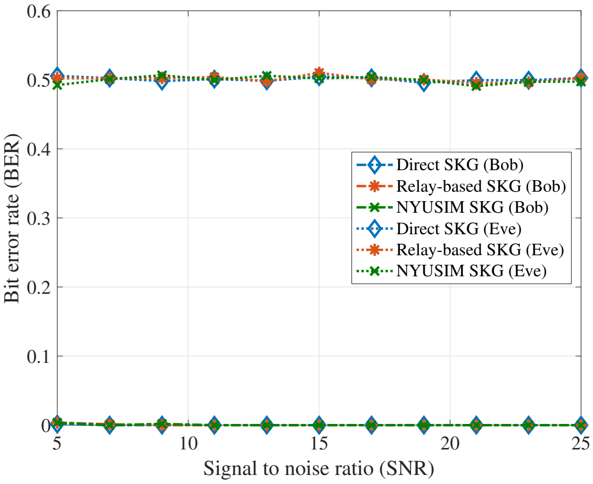


Figure 2.5: The bit error rate (BER) between Alice's sequence and Bob's and Eve's sequences versus SNR.

key sequences. It can be observed that the BER at Bob is extremely low due to the requirement of the consistency checking step in the protocol, which only allows keys whose consistency is verified with a high probability to be accepted. Note that the main reason for the average BER at Eve being around 50% is the privacy amplification step of the protocol. In addition to that, the cumulative distribution function (CDF) of the BER at Eve's final key at 20 dB average SNR for both the direct and NYUSIM-based SKG setups, and 23 dB average SNR for the relay-based SKG setup is shown in Figure 2.6. Note that the curves for all the setups are similar because these curves compare the keys which are the addition modulo 2 of four separate outputs of the hash functions at Alice and Eve. The universal hash function generates a uniformly random output resulting in the similarity of the curves. Also, it is observed that the probability of accepting a mismatched key for the aforementioned average SNRs in the direct and relay-based SKG setups is around 0.0015%, and for the NYUSIM-based SKG setup is around 0.00152%, which are less than 0.00153% as predicted by Theorem 2.2.1. The aforementioned probability is considered to be

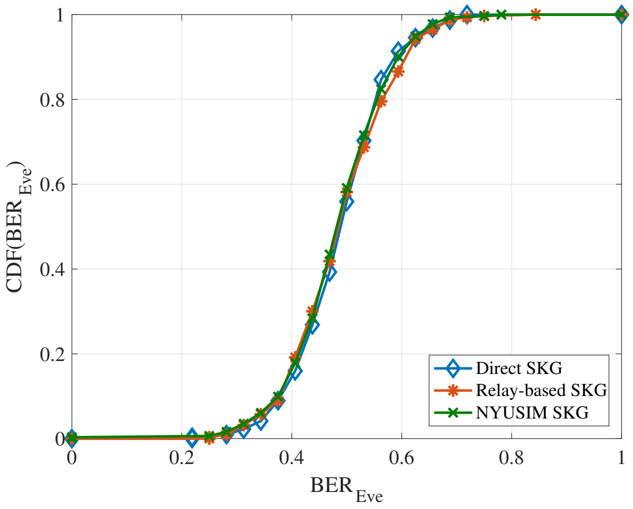


Figure 2.6: The cumulative distribution function of the BER at Eve for the direct, relay-based, and NYUSIM-based SKG setups. The compared sequences are the modulo 2 addition of the outputs of four successful key generation sessions.

very low. In comparison, it is far less than the probability of generating mismatched keys of the protocol proposed for direct SKG in static environments in [62], which is at least 3%. In addition to that, as discussed in the security evaluation of the protocol, the probability of acquiring the key perfectly by Eve is, at most, 2^{-31} and $2^{-10.57}$, for the direct and relay-based SKG, respectively. In comparison, the protocol proposed for direct SKG in static environments in [62] has the probability of acquiring the key by Eve in the range $0.09\% - 0.47\%$.

2.4.2.3 Randomness

The randomness of the generated final key sequence is examined using the NIST statistical test suite [57]. The suite consists of 15 tests and generates a probability value, also referred to as *p*-value, for each individual test. For each test, a sequence is considered random with 99% confidence if the corresponding *p*-value is greater than 0.01. We run the protocol using constant channel coefficients at $20\,\text{dB}$ average SNRs for the direct and NYUSIM-based SKG setups, and $23\,\text{dB}$

Table 2.2: NIST Statistical Test Results

Test	Direct	Relay	NYUSIM
Monobit	0.8712	0.9968	0.2829
Frequency Block	0.3529	0.7458	0.3172
Runs	0.5347	0.9781	0.4516
Longest Run of Ones	0.7696	0.3552	0.1692
Binary Matrix Rank	0.9263	0.9974	0.3125
DFT	0.6413	0.3469	0.0121
Non-Overlapping Template Matching	1	1	1
Overlapping Template Matching	0.2830	0.3501	0.4043
Maurer's Universal Statistical	0.9991	1	0.9993
Linear Complexity	0.9909	0.5323	0.0227
Serial	0.2989	0.5852	0.7236
Approximate Entropy	0.4808	0.9160	0.7529
Cumulative Sums	0.7825	0.8392	0.1833
Random Excursion	0.0179	0.2924	0.0925
Random Excursion Variant Test	0.0434	0.0154	0.0615

average SNR for (2.2.10) in the relay-based SKG setup to generate a sequence of length 2^{20} bits and feed it to the test suite. Since the sequences pass all the tests as shown in Table 2.2, they are considered random with 99% confidence.

2.4.2.4 Randomness Efficiency

This is computed according to (2.4.1). For the direct and NYUSIM-based SKG setups, Alice and Bob randomly choose induced randomness bit sequences of length 64, and therefore, $H(R_a) = H(R_b) = 64$. Note that the length of the quantized bit sequence is 64, therefore, $R_Q = 64$. This implies that the randomness efficiency is 50%. On the other hand, for the relay-based SKG setup, Alice and Bob separately induce 96 random bits during each round, resulting in $H(R_a) = H(R_b) = 96$, while the length of the quantized bit sequence is $R_Q = 64$. The resulting randomness efficiency of the relay-based SKG setup is 33%. Roughly speaking, the remaining part of the available randomness is used to provide security. The exact trade-off between randomness efficiency and security is an interesting problem.

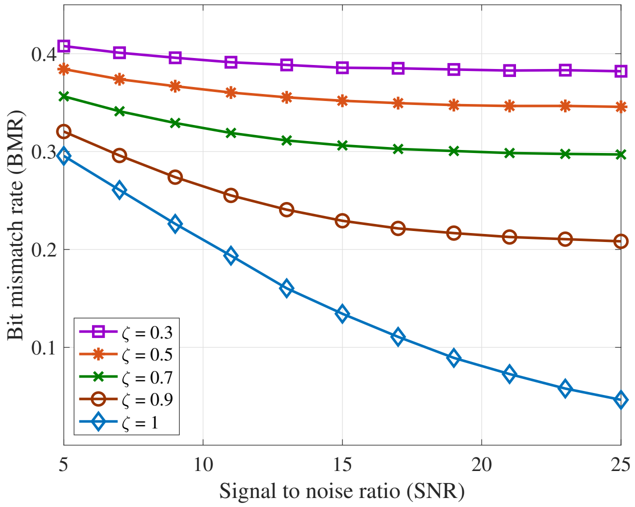


Figure 2.7: The bit mismatch rate (BMR) in the direct SKG setup between Alice's and Bob's sequences versus the signal to noise ratio for different values of the correlation coefficient ζ of the channels experienced at Alice and Bob.

2.4.2.5 Impact of Non-reciprocity

The perfect channel reciprocity feature is assumed to hold throughout the chapter; however, in some practical scenarios, different factors such as mismatched hardware and synchronization errors may cause the channel coefficients experienced at Alice and Bob to not be perfectly reciprocal [60, 63, 64]. Such imperfections can be taken into account using the Pearson correlation coefficient, denoted by ζ, between such channel coefficients explained as follows. In general, under perfect channel reciprocity conditions, we have $\zeta = 1$, while imperfections reduce the value of ζ. As suggested in [64], a model to describe the relationship between the channel coefficients at a subcarrier during session i observed at Alice, i.e., $\widetilde{h}_{ab,i}$, and Bob, i.e., $h_{ab,i}$, when they observe the same SNR is as follows:

$$h_{ab,i} = \zeta \widetilde{h}_{ab,i} + \sqrt{1 - |\zeta|^2} \frac{\sigma_{h_i}}{\sqrt{2}} n_i, \tag{2.4.2}$$

where ζ is the correlation coefficient, $\sigma^2_{h_i}/2$ is the dimension variance of $h_{ab,i}$ and $\widetilde{h}_{ab,i}$, and n_i denotes the circularly-symmetric Gaussian-distributed independent noise component with mean 0 and unit dimension variance. In order to illustrate the effect of imperfect reciprocity in the direct SKG setup, the bit mismatch rate for different values of the correlation coefficient ζ is shown in Figure 2.7. It can be observed that as the correlation coefficient between the channel coefficients decreases, the BMR between Alice's and Bob's quantized sequences increases causing the protocol to experience a higher number of unsuccessful sessions. For instance, to achieve a BMR around 22%, the required SNR is 9 dB for $\zeta = 1$, whereas it is 15 dB for $\zeta = 0.9$. On the other hand, when comparing the average number of sessions required to agree on a key at 15 dB, it is around 9 sessions for $\zeta = 1$, while it is around 37 sessions for $\zeta = 0.9$. Depending on the severity of the imperfections, the protocol's parameters would require certain adjustments to overcome such degradation. For example, the legitimate parties can decrease the bit generation rate by using a lower quantization resolution δ_q, or decrease the rate of the error-correcting code used for reconciliation which results in an increase in the amount of information leaked to the eavesdropper.

2.5 Conclusion

This chapter presented two secret key generation protocols for resource-constrained devices operating in static environments. The protocols utilize the uniqueness of wireless channels, as well as randomness induced at the legitimate parties, to generate highly correlated observations to be used to extract secret key bits. The protocols aim to enable high rate key generation in environments with limited mobility and can be implemented in a low-complexity manner. The protocols' security is evaluated by providing an upper bound of the probability of acquiring the key by an eavesdropper, and their reliability is examined by providing an upper bound of accepting a mismatched key by the legitimate parties. Numerical results are conducted to show the success of the proposed protocols as well as show the applicability of the protocol in scenarios where non-reciprocity is not guaranteed as well as mmWave channel models.

CHAPTER 3

Threshold Secure Coding

3.1 Introduction

This chapter introduces the formulation and code constructions of *threshold*-secure coding with a shared key. Conventional cryptosystems are often designed to be computationally secure by relying on unproven assumptions of hardness of mathematical problems. Information-theoretic security methods provide an alternative approach by constructing codes for keyless secure communication, as in wiretap channels. Typical security schemes aim to secure the entire message block regardless of the deployment setup. We focus on applications where data is generated such that an eavesdropper with partial knowledge of the message cannot deduce any meaningful information about the message. In other words, the eavesdropper needs to retrieve most or even the entire block of message symbols to learn meaningful information about the message. We also aim to combine the processes of security and error correction in the physical layer. Utilizing error-correcting codes to provide security in the physical layer enables sharing hardware resources between reliability and security schemes in low-cost devices.

In this chapter, we formulate the threshold security conditions using information-theoretic metrics for noiseless communication channels. Furthermore, we propose a general construction of

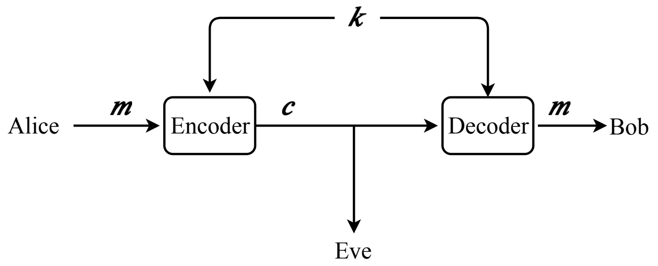


Figure 3.1: System setup for the proposed coding scheme.

threshold-secure coding schemes based on linear block codes and show the threshold achievable by such construction is directly related to the minimum distance of the linear block code. Moreover, the setup is extended to the case where the main channel is noisy, but the eavesdropper's channel remains noiseless for which a robust coding scheme is also presented. For both setups, we provide low-complexity constructions of threshold-secure and robust coding schemes based on Reed-Muller codes.

3.2 Problem Formulation

We start with a simple setup in terms of communication channels to highlight the threshold security problem. Consider a system model where Alice wishes to securely communicate with Bob, both are legitimate parties, through a noiseless channel. The eavesdropper, namely Eve, is tapping into that channel and observes all the transmitted symbols, as shown in Figure 3.1. Alice and Bob share a common key sequence $\boldsymbol{k}$ of length k, that can be used for encoding and decoding of the message $\boldsymbol{m}$ of length m. Assume that both the key and the message symbols are from an alphabet of size q, where q is a prime power. A certain known permutation $\pi(\cdot)$ of Alice's message sequence $\boldsymbol{m}$ together with the key sequence $\boldsymbol{k}$ is fed as the input to the encoder, denoted by $\boldsymbol{u}$, i.e., $\boldsymbol{u} = \pi(\boldsymbol{k}, \boldsymbol{m})$. The length of $\boldsymbol{u}$ is $n = m + k$ and is encoded to a codeword $\boldsymbol{c}$ of length m. The entries in $\boldsymbol{k}$, as well as $\boldsymbol{m}$, are assumed to be independent and uniformly distributed. The codeword $\boldsymbol{c}$ is then transmitted by Alice to Bob over a noiseless channel. Bob receives the codeword and decodes it using the key $\boldsymbol{k}$ to retrieve the message $\boldsymbol{m}$. Eve observes the codeword $\boldsymbol{c}$ and wishes to extract information about the message $\boldsymbol{m}$ as well as the key $\boldsymbol{k}$. It is worth noting that Alice and Bob agree on the encoder and the decoder a priori, which are also publicly known to Eve.

The security condition in this setup is the following. Although parts of input $\boldsymbol{u}$ are disclosed to Eve, no knowledge, in an information-theoretic sense, about any subset of size up to a certain threshold parameter t of the input symbols is leaked to Eve. In a sense, we consider a *sub-block-wise* measure of information-theoretic security. We aim at designing an encoder and a decoder for a noiseless channel that utilize a shared key $\boldsymbol{k}$ to encode and decode a message $\boldsymbol{m}$ such that the following conditions are met:

1. *Reliability*: Bob is able to decode the message, knowing the key, with probability one, i.e.,

$$H(\boldsymbol{m}|\boldsymbol{c},\boldsymbol{k}) = 0. \tag{3.2.1}$$

2. *Key security*: the codeword $\boldsymbol{c}$ does not reveal any information about the key $\boldsymbol{k}$, i.e.,

$$I(\boldsymbol{k};\boldsymbol{c}) = 0. \tag{3.2.2}$$

3. t*-threshold security*: for any $\boldsymbol{v} \subseteq \{u_1,u_2,..., u_n\}$ with $|\boldsymbol{v}| \leqslant t$, we have

$$H(\boldsymbol{v}|\boldsymbol{c}) = H(\boldsymbol{v}), \tag{3.2.3}$$

where t is a design parameter specified later.

A formal definition of a t-threshold secure code is stated next.

Definition 3.1 (threshold-secure code) *A code is said to be t-threshold secure if it meets the reliability and security conditions, where t is the maximum cardinality of any $\boldsymbol{v} \subseteq \{u_1, u_2, ..., u_n\}$ that satisfies* (3.2.3).

3.2.1 Applications

The considered threshold-type security becomes relevant in applications where the entire message, or significant portion of it, is needed in order for an eavesdropper to obtain meaningful knowledge about the content of the message. Next, we briefly expand on one of the applications for the described threshold security setup.

Consider an authentication system based on users' biometric information, such as fingerprints, e.g., as described in [65], where the data is assumed to be hashed prior to encoding. Let us denote the fingerprint measurement vector as $\tilde{\boldsymbol{x}}$. Also, let us have the following two functions: a feature extraction function $g(\cdot)$ and a secure hash function $h(\cdot)$. The function $g(\cdot)$ is an arbitrary function that maps the input vector $\tilde{\boldsymbol{x}}$ to another vector $\boldsymbol{x}$. The hash function $h(\cdot)$ is a mapping from an input space of size a to a hash table of size b with the following property:

$$\Pr(h(\boldsymbol{x}_1) = h(\boldsymbol{x}_2) | \boldsymbol{x}_1 \neq \boldsymbol{x}_2) = \frac{1}{b}, \tag{3.2.4}$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are any input vectors, and the resulting load factor of this hash function is $\beta_l = \frac{a}{b}$ [66]. In this example, when a user scans their fingerprint, the measurement vector $\tilde{\boldsymbol{x}}$ is processed using $g(\cdot)$ to produce the vector $\boldsymbol{x}$ that is hashed using the hash function $h(\cdot)$ to produce the hashed vector denoted as $\boldsymbol{m}$, i.e.,

$$\boldsymbol{m} = h(\boldsymbol{x}) = h(g(\tilde{\boldsymbol{x}})). \tag{3.2.5}$$

Then the hashed vector $\boldsymbol{m}$ is the input to the threshold-secure encoder together with the key. This hashed vector is uniformly distributed by the assumption on the hash function $h(\cdot)$ in (3.2.4). The hashed vector is to be sent to a database that contains the hashed vectors of all authorized users for authentication. For an eavesdropper that aims to learn the vector $\boldsymbol{x}$, knowledge of the entire $\boldsymbol{m}$ is needed. Let us assume that the eavesdropper has access to the hash function $h(\cdot)$. If $\boldsymbol{m}$ is sent as is, the probability of successfully acquiring $\boldsymbol{x}$ by the eavesdropper is $\frac{1}{\beta_l}$ since the eavesdropper can discard any vector that does not hash to the observed $\boldsymbol{m}$. However, when using threshold-secure coding with threshold t, and assuming an alphabet of size q, this probability becomes at most $\frac{1}{\beta_l q^t}$ which is exponentially decaying with t. This is because the eavesdropper needs to retrieve the hashed vector $\boldsymbol{m}$ first. Choosing an appropriate parameter t, e.g., in the order of a few tens, combined with the uniformity of the hash functions, is sufficient to cripple the eavesdropper in a practical setting.

3.3 Proposed Coding Scheme

With a slight abuse of terminology, we refer to a scheme meeting the reliability and security conditions, as described in the previous section, simply as a coding scheme. The coding scheme is revealed to all parties, i.e., Alice, Bob, and Eve. When constructing the coding scheme, we aim at designing an encoder and a decoder as well as specifying the code. For an input vector $\boldsymbol{u} = \pi(\boldsymbol{k}, \boldsymbol{m})$, the encoder produces a codeword $\boldsymbol{c}$ as follows

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{H} = \pi(\boldsymbol{k}, \boldsymbol{m})\boldsymbol{H}, \tag{3.3.1}$$

where $\boldsymbol{H}$ is an $n \times m$ matrix with $n = m + k$. In our proposed scheme, the matrix $\boldsymbol{H}$ is chosen as the transpose of a generator matrix $\boldsymbol{G}$ of a linear block code.

Consider a $[n, m, d_{\min}]_q$ linear block code with generator matrix $\boldsymbol{G}$, i.e., a linear block code whose elements are from an alphabet of size q, and has rate $R = m/n$ and minimum distance $d_{\min}$. We aim at utilizing the generator matrix $\boldsymbol{G}$ of certain linear block codes to construct a matrix $\boldsymbol{H}$ for our coding scheme such that the reliability and security conditions are met.

One can assume that the length of the key is less than the length of the message. To encode a message $\boldsymbol{m}$, let us denote the set of indices of the rows of $\boldsymbol{H}$ that correspond to the message symbols as $\mathcal{A} \subseteq [m + k] \overset{\text{def}}{=} \{1, 2, ..., m + k\}$. Then the set of indices of the rows corresponding to the key symbols is $\mathcal{A}^c = [m + k] \setminus \mathcal{A}$. The matrix $\boldsymbol{H}_{\mathcal{A}}$ denotes the submatrix of $\boldsymbol{H}$ with rows indexed by $\mathcal{A}$, and the matrix $\boldsymbol{H}_{\mathcal{A}^c}$ denotes the submatrix of $\boldsymbol{H}$ with rows indexed by $\mathcal{A}^c$. The codeword $\boldsymbol{c}$ is then expressed as follows

$$\boldsymbol{c} = \boldsymbol{m}\boldsymbol{H}_{\mathcal{A}} + \boldsymbol{k}\boldsymbol{H}_{\mathcal{A}^c}. \tag{3.3.2}$$

The choice of $\pi(\cdot)$, which corresponds to the choice of $\mathcal{A}$ and $\mathcal{A}^c$, is critical in ensuring security and reliability conditions. Hence, we have the following definition.

Definition 3.2 (proper codes) *A code, as described above, is called* proper *if its matrix satisfies* *the following requirements:*

1. *The resulting submatrix* $\mathbf{H}_{\mathcal{A}}$ *is full row rank, i.e.,* $\operatorname{rank}(\boldsymbol{H}_{\mathcal{A}}) = m$.

2. *The resulting submatrix* $\mathbf{H}_{\mathcal{A}^c}$ *is also full row rank, i.e.,* $\operatorname{rank}(\boldsymbol{H}_{\mathcal{A}^c}) = k$.

As will be shown throughout the rest of this section, a code that is not *proper* will result in a lower equivocation rate for Eve about the message and leads to leakage of information about the key to Eve.

Next, we show that if a code is *proper*, then it meets the reliability condition, as specified in (3.2.1), and the security conditions, as specified in (3.2.2) and (3.2.3). The following lemma shows that the reliability condition is satisfied.

Lemma 3.3.1 *Suppose that the code used in the coding scheme is* proper*, as defined in Definition 3.2. Then Bob can recover the message with probability one under maximum a posteriori (MAP) decoding. In other words,*

$$H(\boldsymbol{m}|\boldsymbol{c}, \boldsymbol{k}) = 0. \tag{3.3.3}$$

Proof: By using (3.3.2), it can be observed that since Bob has $\boldsymbol{c}$ and $\boldsymbol{k}$ and since $\boldsymbol{H}_{\mathcal{A}}$ is full rank, then Bob can subtract $\boldsymbol{k}\boldsymbol{H}_{\mathcal{A}^c}$ from $\boldsymbol{c}$ and then retrieve $\boldsymbol{m}$ using $\boldsymbol{H}_{\mathcal{A}}$, which has a unique solution. ■

Next, we show that a *proper* code meets the key security condition, as specified in (3.2.2). Note that satisfying this condition is critical as even a very small leakage of the key $\boldsymbol{k}$ can lead to compromising the security of the message.

Theorem 3.3.2 *Suppose that the code used in the coding scheme is* proper*, as defined in Definition 3.2. Then the codeword* **c** *leaks no information about the key* **k***, i.e.,*

$$I(\boldsymbol{k};\boldsymbol{c}) = 0. \tag{3.3.4}$$

Proof: The proof is by observing the following set of equalities:

$$\begin{aligned} I(\boldsymbol{k};\boldsymbol{c}) &= H(\boldsymbol{c}) - H(\boldsymbol{c}|\boldsymbol{k}), && (3.3.5)\\ &= m\log_2(q) - H(\boldsymbol{m}\boldsymbol{H}_{\mathcal{A}} + \boldsymbol{k}\boldsymbol{H}_{\mathcal{A}^c}|\boldsymbol{k}), && (3.3.6)\\ &= m\log_2(q) - H(\boldsymbol{m}\boldsymbol{H}_{\mathcal{A}}), && (3.3.7)\\ &= \log_2(q)(m - \mathrm{rank}(\boldsymbol{H}_{\mathcal{A}})), && (3.3.8)\\ &= 0, && (3.3.9) \end{aligned}$$

where (3.3.6) holds by (3.3.2) and the uniformity of the key and message symbols, hence the codewords are uniform, (3.3.7) holds because $\boldsymbol{m}$ and $\boldsymbol{k}$ are independent, (3.3.8) is by noting that elements of $\boldsymbol{m}$ are uniformly distributed and independent, and (3.3.9) holds because $\mathrm{rank}(\boldsymbol{H}_{\mathcal{A}}) = m$ as the code is *proper* according to Definition 3.2. ■

The following lemma is well-known. However, it is included here as it is instrumental in characterizing the threshold security of coding schemes based on linear block codes.

Lemma 3.3.3 *[67] For an* $[n, m, d_{\min}]_q$ *linear block code with generator matrix* **G***, any submatrix of* **G** *of size* $m \times (n - |\mathcal{E}|)$ *obtained by removing columns indexed by elements of* $\mathcal{E}$*, where* $\mathcal{E} \subseteq [n]$

with $|\mathcal{E}| = d_{\min} - 1$, *has full row rank, i.e.,*

$$\operatorname{rank}(\boldsymbol{G}_{\mathcal{E}^c}) = m. \tag{3.3.10}$$

Characterizing the threshold security of coding schemes based on linear block codes follows.

Theorem 3.3.4 *A coding scheme constructed by a matrix* $\boldsymbol{H} = \boldsymbol{G}^T$, *where* $\boldsymbol{G}$ *is the generator matrix of an* $[n, m, d_{\min}]_q$ *linear block code, is* t*-threshold secure, where* $t = d_{\min} - 1$, *i.e., we have*

$$H(\boldsymbol{v}|\boldsymbol{c}) = H(\boldsymbol{v}), \tag{3.3.11}$$

for any $\boldsymbol{v} \subseteq \{u_1, u_2, ..., u_n\}$ *with* $|\boldsymbol{v}| = t$, *and* t *is the maximum value for which this condition holds.*

Proof: Let $\boldsymbol{u}$ denote the input to the encoder for the coding scheme, as specified in (3.3.1). Suppose that $\boldsymbol{v}$ consists of elements of $\boldsymbol{u}$ indexed by $\mathcal{B} = \{i_1, i_2, ..., i_t\} \subseteq [n]$, and $\tilde{\boldsymbol{v}}$ consists of elements of $\boldsymbol{u}$ indexed by $\mathcal{B}^c = [n] \setminus \mathcal{B}$. Then we have the following:

$$I(\boldsymbol{v}; \boldsymbol{c}) = H(\boldsymbol{c}) - H(\boldsymbol{c}|\boldsymbol{v}), \tag{3.3.12}$$

$$= m \log_2(q) - H(\tilde{\boldsymbol{v}}\boldsymbol{H}_{\mathcal{B}^c} + \boldsymbol{v}\boldsymbol{H}_{\mathcal{B}}|\boldsymbol{v}), \tag{3.3.13}$$

$$= m \log_2(q) - H(\tilde{\boldsymbol{v}}\boldsymbol{H}_{\mathcal{B}^c}), \tag{3.3.14}$$

$$= \log_2(q)(m - \operatorname{rank}(\boldsymbol{H}_{\mathcal{B}^c})), \tag{3.3.15}$$

$$= 0, \tag{3.3.16}$$

where (3.3.13) follows due to codewords being uniformly distributed and expansion of random variables, (3.3.14) holds by the independence of $\boldsymbol{v}$ and $\tilde{\boldsymbol{v}}$, (3.3.15) holds due to the uniformity of $\tilde{\boldsymbol{v}}$, and (3.3.16) holds by Lemma 3.3.3 with $t = d_{\min} - 1$. Since the mutual information $I(\boldsymbol{v}; \boldsymbol{c})$ is zero, it implies that the t-threshold security criteria is met for the parameter $t = d_{\min} - 1$, i.e.,

$$H(\boldsymbol{v}|\boldsymbol{c}) = H(\boldsymbol{v}), \tag{3.3.17}$$

for any $\boldsymbol{v}$ with $|\boldsymbol{v}| = t$, where $t = d_{\min} - 1$.

Next, we need to show that $t = d_{\min} - 1$ is the maximum value for which the threshold security condition holds. Consider a codeword in the codebook generated by $\boldsymbol{G}$ that has the Hamming weight equal to $t + 1 = d_{\min}$ with non-zero elements at indices denoted by $\mathcal{G} = \{i_1, i_2, ..., i_{t+1}\}$. Then we have the following:

$$\begin{aligned} H(u_{i_1}, ..., u_{i_{t+1}}|\boldsymbol{c}) &= H(u_{i_1}, ..., u_{i_t}|\boldsymbol{c}) + H(u_{i_{t+1}}|\boldsymbol{c}, u_{i_1}, ..., u_{i_t}), && (3.3.18)\\ &= H(u_{i_1}, ..., u_{i_t}|\boldsymbol{c}), && (3.3.19)\\ &\neq H(u_{i_1}, ..., u_{i_{t+1}}), && (3.3.20) \end{aligned}$$

where (3.3.18) follows from the chain rule of entropy, and (3.3.19) holds because there exists a linear combination of the entries of $\boldsymbol{c} = (c_1, c_2, ..., c_m)$ such that $\sum_{i \in [m]} \iota_i c_i = \sum_{j \in \mathcal{G}} o_j u_j$. Hence, the second term becomes zero, since $u_{i_{t+1}}$ is uniquely determined given $\boldsymbol{c}$ and $\{u_{i_1}, ..., u_{i_t}\}$. Therefore, due to (3.3.20), the threshold security condition does not hold for $t + 1 = d_{\min}$. ■

Maximizing the threshold of the coding scheme is directly related to maximizing the minimum distance of the linear block code. This provides an upper bound on the achievable threshold, i.e., $t \leqslant k$. The next theorem states there exists a coding scheme achieving the maximum threshold.

Theorem 3.3.5 *For any message length m and key length k, there exists a* proper *code with threshold $t = k$, provided that the alphabet size $q \geqslant m + k + 1$.*

Proof: To prove the theorem, we give an example of a code that is shown to be *proper* with $t = k$. We utilize Reed-Solomon (RS) codes, which are a well-known family of codes that are maximum distance separable (MDS) codes, i.e., $d_{\min} = n - m + 1 = k + 1$ [67]. For any $[n, m, d_{\min}]_q$ RS code, all we need to show is that the matrix $\boldsymbol{H}$ which is the transpose of the generator matrix $\boldsymbol{G}$ of the RS code can be used to construct a *proper* code. One of the properties of MDS codes is that every set of m columns of the matrix $\boldsymbol{G}$ are linearly independent [67, Proposition 11.4]. Note that rows of $\boldsymbol{H}$ correspond to columns of $\boldsymbol{G}$. Hence, any choice of m columns of $\boldsymbol{G}$ will have rank m, and the remaining k columns of $\boldsymbol{G}$ will also have rank k as it is assumed that $k < m$. Therefore, the code generated by $\boldsymbol{H}$ is *proper*, with threshold $t = k$. ■

3.4 Low Complexity Construction

This section focuses on designing binary codes that meet the reliability and security conditions while providing encoding and decoding algorithms with linear/quasi-linear complexity. To this end, we consider Reed-Muller codes due to their recursive construction and low-complexity decoder. In addition, since they are designed with the objective of maximizing the minimum distance, we can achieve a reasonably high threshold t for the t-threshold security condition.

3.4.1 Encoder

We start first by describing Reed-Muller codes. An RM(s, r) code is a $[2^s, \sum_{i=0}^{r} \binom{s}{i}, 2^{s-r}]_2$ linear block code. The generator matrix of the RM(s, r) code, denoted by $\boldsymbol{G}(s, r)$, is obtained by keeping the rows with the Hamming weight of at least 2^{s-r} from the matrix $\boldsymbol{S}^T = (\boldsymbol{S}_2^{\otimes s})^T$ and removing the remaining rows, where $\otimes$ denotes the Kronecker product, T is the transpose operator, and $\boldsymbol{S}_2$ is the following kernel matrix

$$\boldsymbol{S}_2 = \begin{bmatrix} 1 & 0 \\ \\ 1 & 1 \end{bmatrix}. \tag{3.4.1}$$

This is one of the methods to construct RM codes. This method is chosen so that we can describe a code construction that satisfies the *proper* condition of threshold-secure codes. Due to the recursive structure of $\boldsymbol{S}$, it can be observed that indices of the rows with the lowest weight, the second-lowest weight, etc, from $\boldsymbol{S}$ correspond to indices of columns with the highest column weight, the second-highest weight, etc, from $\boldsymbol{S}$, respectively. When specifying the matrix $\boldsymbol{G}(s, r)$ as a sub-matrix of $\boldsymbol{S}^T$ we choose the set of indices of the removed rows from $\boldsymbol{S}^T$ as $\mathcal{A}^c$ to assign the rows of $\boldsymbol{H}$ dedicated for the key, while the indices of the remaining rows are used as the message indices $\mathcal{A}$.

In the proposed scheme based on RM codes, we have $n = 2^s$, $m = \sum_{i=0}^{r} \binom{s}{i}$, for some $r \geqslant s/2$, and $k = n - m < m$. Note that the underlying RM code has rate $R > \frac{1}{2}$. By using Theorem 3.3.4 and noting that the minimum distance of the underlying code is 2^{s-r}, the achievable threshold security parameter t for the RM-based scheme with parameters (s, r) is $t = 2^{s-r} - 1$. Note that, in general, for an RM code of constant rate, i.e., $R = O(1)$, we have $r = s/2 + O(\sqrt{s})$. Hence, the threshold security parameter of the corresponding scheme is $t = \sqrt{n}\exp(O(\sqrt{\log n}))$.

Next, we show the proposed construction results in a *proper* code.

Proposition 3.4.1 *The choice of the sets $\mathcal{A}$, and $\mathcal{A}^c$ as mentioned above results in a* proper *code.*

Proof: To prove this proposition, it suffices to show that $\boldsymbol{H}_{\mathcal{A}}$ and $\boldsymbol{H}_{\mathcal{A}^c}$ are both full row rank.

First, it is shown that $\boldsymbol{H}_{\mathcal{A}}$ is full row rank. Note that for a full rank lower-triangular matrix, a submatrix obtained by removing a subset of columns and rows with the same indices results also in a full rank lower-triangular matrix. Also, note that $\mathcal{A}^c$ is the subset of indices of deleted columns as well as that of the rows dedicated for the key from $\boldsymbol{S}$. Hence, the matrix $\boldsymbol{H}_{\mathcal{A}}$ is full row rank.

Next, we show that $\boldsymbol{H}_{\mathcal{A}^c}$ is full row rank. This is done by induction. Note that $k < m$ is assumed, as mentioned before. Also, to simplify the proof, let us have $r' = s - r$, and also re-express k and m in the remainder of the proof as follows

$$k = \sum_{i=0}^{r'} \binom{s}{i},$$

and

$$m = \sum_{i=r'+1}^{s} \binom{s}{i},$$

where we have $r' \leqslant \left\lfloor \frac{s-1}{2} \right\rfloor$. Note that $\boldsymbol{H}_{\mathcal{A}^c}$ contains the $\sum_{i=0}^{r'} \binom{s}{i}$ rows dedicated for the key from $\boldsymbol{S}$ with the same number of lowest-weight columns removed. Let this matrix be also denoted by $\boldsymbol{S}(s, r')$. Let also $\boldsymbol{S}'(s, r')$ denote the matrix that contains the $\sum_{i=0}^{r'} \binom{s}{i}$ rows dedicated for the key from $\boldsymbol{S}$ with only $\sum_{i=0}^{r'-1} \binom{s}{i}$ lowest weight columns removed. Due to the recursive structure of the matrix $\boldsymbol{S}$, $\boldsymbol{S}(s, r')$ can be expressed as follows:

$$\boldsymbol{S}(s,r') = \begin{bmatrix} \boldsymbol{S}(s-1,r'-1) & \boldsymbol{0} \\ \boldsymbol{S}'(s-1,r') & \boldsymbol{S}(s-1,r') \end{bmatrix}. \tag{3.4.2}$$

Next, we show that the matrix $\boldsymbol{S}(s,r')$ is full row rank for the maximum value $r' = \left\lfloor \frac{s-1}{2} \right\rfloor$ and for $s \geqslant 2$ by induction on s. Then it will be discussed why this also holds for $r' < \left\lfloor \frac{s-1}{2} \right\rfloor$.

Step 1: The induction basis is for $s = 2$ and $r' = 0$, and for $s = 3$ and $r' = 1$, which can be easily verified, i.e., for $s = 2$ and $r' = 0$, the rank of $\boldsymbol{S}(2,0)$ is 1. Also, for $s = 3$ and $r' = 1$, the rank of $\boldsymbol{S}(3,1)$ is 4.

Step 2: Suppose that the induction hypothesis holds for s and s is odd. Then we have the following matrix:

$$\boldsymbol{S}(s+1,r') = \begin{bmatrix} \boldsymbol{S}(s,r'-1) & \boldsymbol{0} \\ \boldsymbol{S}'(s,r') & \boldsymbol{S}(s,r') \end{bmatrix}. \tag{3.4.3}$$

We need to show that $\mathrm{rank}(\boldsymbol{S}(s+1,r')) = \sum_{i=0}^{r'} \binom{s+1}{i}$. Note that $\boldsymbol{S}(s,r')$ is full row rank by induction hypothesis, i.e., $\mathrm{rank}(\boldsymbol{S}(s,r')) = \sum_{i=0}^{r'} \binom{s}{i}$. Then $\boldsymbol{S}(s,r'-1)$, which contains a subset of the rows in $\boldsymbol{S}(s,r')$, is also full row rank. Hence, we have $\mathrm{rank}(\boldsymbol{S}(s,r'-1)) = \sum_{i=0}^{r'-1} \binom{s}{i}$.

Therefore,

$$\begin{aligned}\operatorname{rank}(\boldsymbol{S}(s+1,r')) &= \operatorname{rank}(\boldsymbol{S}(s,r'-1)) + \operatorname{rank}(\boldsymbol{S}(s,r')), && (3.4.4)\\ &= \sum_{i=0}^{r'-1}\binom{s}{i} + \sum_{i=0}^{r'}\binom{s}{i}, && (3.4.5)\\ &= \sum_{i=0}^{r'}\binom{s+1}{i}, && (3.4.6)\end{aligned}$$

which is equal to the number of rows in $\boldsymbol{S}(s+1,r')$. Hence, it is full row rank.

For even s with corresponding parameter r', we need to show the following matrix is full row rank

$$\boldsymbol{S}(s+1,r'+1) = \begin{bmatrix} \boldsymbol{S}(s,r') & \boldsymbol{0} \\ \boldsymbol{S}'(s,r'+1) & \boldsymbol{S}(s,r'+1) \end{bmatrix}. \qquad (3.4.7)$$

First, we have $\operatorname{rank}(\boldsymbol{S}(s,r')) = \sum_{i=0}^{r'}\binom{s}{i}$ by induction hypothesis. Regarding $\operatorname{rank}(\boldsymbol{S}'(s,r'+1))$, we can see that $\boldsymbol{S}'(s,r'+1)$ has $\sum_{i=0}^{r'}\binom{s}{i}$ rows that are also included in $\boldsymbol{S}(s,r')$. However, when considering the indices of such rows in $[\boldsymbol{S}'(s,r'+1)\ \boldsymbol{S}(s,r'+1)]$, the corresponding rows are independent from all other rows in $[\boldsymbol{S}(s,r'),\ \boldsymbol{0}]$. Furthermore, there are $\binom{s}{r'+1}$ additional rows in $\boldsymbol{S}'(s,r'+1)$ that are linearly independent from the remaining rows due to the structure of the zero blocks in this matrix, similar to (3.4.2). We can then compute the rank of $\boldsymbol{S}(s+1, r'+1)$ as follows

$$\begin{aligned}
\operatorname{rank}(\boldsymbol{S}(s+1, r'+1)) &= \operatorname{rank}(\boldsymbol{S}(s, r')) + \operatorname{rank}(\boldsymbol{S}'(s, r'+1)), && (3.4.8)\\
&= \sum_{i=0}^{r'} \binom{s}{i} + \sum_{i=0}^{r'} \binom{s}{i} + \binom{s}{r'+1}, && (3.4.9)\\
&= \sum_{i=0}^{r'+1} \binom{s+1}{i}. && (3.4.10)
\end{aligned}$$

Hence, $\boldsymbol{S}(s+1, r'+1)$ is full row rank, and the induction hypothesis holds for $s+1$ with the maximum value of r'. For keys of shorter lengths, it is straightforward to see that for any $r'' < r'$, the matrix $\boldsymbol{S}(s, r'')$ whose rows are a subset of $\boldsymbol{S}(s, r')$ with additional columns inserted at different locations is also full row rank. This completes the proof. ■

3.4.2 Decoder

In this part, we discuss a low-complexity successive cancellation (SC) decoder to decode the message in the RM-based coding scheme while utilizing the shared key. As Reed-Muller codes are closely related to polar codes [68], a decoder closely related to that of polar codes described in [68] is natural. However, there are fundamental differences that will be clarified throughout this section.

The decoder is described in Algorithm 3.1. We first embed erasures within the entries of the codeword $\boldsymbol{c}$ in order to get a vector of length n, denoted by $\bar{\boldsymbol{c}}$, by inserting the erasures at locations indexed by $\mathcal{A}^c$. More specifically, $\bar{\boldsymbol{c}} = \pi_1(\boldsymbol{e}_k, \boldsymbol{c})$ where $\boldsymbol{c}$ is the codeword and $\boldsymbol{e}_k$ is an erasure vector of length k such that the permutation places the erasures at locations denoted by $\mathcal{A}^c$. Note that, as mentioned before, $\mathcal{A}^c$ corresponds to the location of the key bits at the encoder.

The decoder takes the key bits $\boldsymbol{k}$, the codeword embedded with erasures $\bar{\boldsymbol{c}} = \pi_1(\boldsymbol{e}_k, \boldsymbol{c})$, indices of the key bits $\mathcal{A}^c$ and a recursion index j as inputs, and outputs the vector $\boldsymbol{u} = [u_1, u_2, ..., u_n] = \pi(\boldsymbol{k}, \boldsymbol{m})$ from which the message can be retrieved $\boldsymbol{m} = \boldsymbol{u}_{\mathcal{A}}$. The high-level idea of the decoder is as follows. The vector $\bar{\boldsymbol{c}}$ is divided into two parts; $\bar{c}_1^{n/2} = [\bar{c}_1, \bar{c}_2, ..., \bar{c}_{n/2}]$ and $\bar{c}_{n/2+1}^{n} = [\bar{c}_{n/2+1}, \bar{c}_{n/2+2}, ..., \bar{c}_n]$, that are decoded successively. As opposed to the SC decoder of polar codes [68], the second sub-block is processed first, cancelled from the first sub-block, and then the first sub-block is processed. Each of these sub-blocks is also decoded recursively by splitting them into two parts and so on.

Remark. When describing the recursive SC decoding process we often use the binary tree terminology in which the input codeword, i.e., $\bar{\boldsymbol{c}}$, is assigned to the root of the tree and then the first and the second sub-blocks are assigned to the *left child* and the *right child*, respectively. The decisions are made at the leaves of the tree and then are re-encoded and propagated back through the tree, see, e.g., [69] for more details.

The following claim verifies that the decoder successfully outputs the message bits with probability 1 for any key length. Note that since the proof follows by induction, we discard the assumption that $k \leqslant m$ and simply show the claim for any $k \leqslant n$.

Claim 3.4.2 *The RM-based coding scheme can be successfully decoded using the SC decoder in Algorithm 3.1 for any key length $k \leqslant n$.*

Proof: We use induction on l, where $n = 2^l$, to show that the claim holds.

Step 1: For the induction basis, consider $n = 2$. We need to show decoding is successful for $k = 0, 1, 2$. For $k = 0$, which corresponds to the case with no erasure, the induction hypothesis holds trivially as $\boldsymbol{S}$ is non-singular. For $k = 1$, one needs to show the induction hypothesis for both

Algorithm 3.1 Successive cancellation decoder (Decoder)

```
Initialization: $j = 1$.
Input: $\boldsymbol{k}$, $\bar{c}_1^n = \pi_1(\boldsymbol{e}_k, \boldsymbol{c})$, $\mathcal{A}^c$, $j$.
Output: $b_1^n$, $u_1^n$.
**if** $n = 2$ **then**
    **if** $\bar{c}_2 = e$ **then**
        $u_j = k_j$
    **else**
        $u_j = \bar{c}_2$
    **end if**
    **if** $\bar{c}_1 = e$ **then**
        $u_{j-1} = k_{j-1}$
    **else**
        $u_{j-1} = u_j \oplus \bar{c}_1$
    **end if**
    $b_1^n = [u_{j-1} \oplus u_j, u_j]$
**else**
    $\boldsymbol{b}' \leftarrow$ Decoder($\boldsymbol{k}_2, \bar{c}^n_{n/2+1}, \mathcal{A}^c_2, 2j$)
    $\tilde{c}_1^{n/2} = \boldsymbol{b}' \oplus \bar{c}_1^{n/2}$
    $\boldsymbol{b}'' \leftarrow$ Decoder($\boldsymbol{k}_1, \tilde{c}_1^{n/2}, \mathcal{A}^c_1, 2j - 1$)
    $b_1^n = [\boldsymbol{b}'' \oplus \boldsymbol{b}', \boldsymbol{b}']$
**end if**
**return** $b_1^n$
```

possible cases for $\mathcal{A}^c$. First, let us consider that $\bar{c}_1 = e$ and $\bar{c}_2 = c_1$, which corresponds to $u_1 = k_1$, and $u_2 = m_1$. In this case, the decoder outputs $u_1 = k_1$ and $u_2 = \bar{c}_2$. For the other case where $\bar{c}_1 = c_1$ and $\bar{c}_2 = e$, which corresponds to $u_1 = m_1$ and $u_2 = k_1$, the decoder first corrects the erasure, assigning $u_2 = k_1$. It then computes $u_1 = m_1 = u_2 \oplus \bar{c}_1 = k_1 \oplus \bar{c}_1$. Finally, we show it succeeds for $k = 2$, where both $\bar{c}_1$ and $\bar{c}_2$ are erased. Then $u_1 = k_1$ and $u_2 = k_2$ and the decoder is successful.

Step 2: Now, suppose that the induction hypothesis holds for $n = 2^l$ and for any $k \leqslant 2^l$, where k is the length of the key, regardless of the indices of the key bits. However, note that, as specified before, the row indices corresponding to the key bits and the column indices corresponding to the erasures are the same and are both denoted by $\mathcal{A}^c$. We now show that the claim is true for $n = 2^{l+1}$

and any $k \leqslant 2^{l+1}$. Let us split the key indices $\mathcal{A}^c$ into two sets, $\mathcal{A}_1^c$ and $\mathcal{A}_2^c$, with sizes $|\mathcal{A}_1^c| = k_1$ and $|\mathcal{A}_2^c| = k_2$, where $k = k_1 + k_2$, as follows. The set $\mathcal{A}_1^c$ consists of the indices of erasures in $\bar{c}_1^{n/2}$. Also, let $\boldsymbol{k}_1$ denote the corresponding part of the key of size k_1. Similarly, $\mathcal{A}_2^c$ consists of the indices of erasures in $\bar{c}_{n/2+1}^n$. Also, let $\boldsymbol{k}_2$ denote the corresponding part of the key of size k_2. First, the right child with input $\bar{c}_{n/2+1}^n$, which has k_2 erasures, is processed. Note that there are also k_2 known key bits indexed by $\mathcal{A}_2^c$ in the second half sub-block $u_{n/2+1}^n$. Note that the decoder for the right child has an input of length $n' = 2^l$ and $k' = k_2$ erasures as well as key bits $\boldsymbol{k}_2$ indexed by $\mathcal{A}_2^c$. The decoder succeeds by the induction hypothesis. The right child then passes

$$u_{n/2+1}^n \boldsymbol{S}_2^{\otimes l} \oplus \bar{c}_1^{n/2} = \boldsymbol{b}' \oplus \bar{c}_1^{n/2} = \tilde{c}_1^{n/2}$$

to the left child. The decoder is then run on $\tilde{c}_1^{n/2}$, which is of length $n' = 2^l$ and has $k' = k_1$ erasures and key bits $\boldsymbol{k}_1$ indexed by $\mathcal{A}_1^c$. The decoder is successful on this node as well by the induction hypothesis. Hence, the decoder is successful for $n = 2^{l+1}$ which completes the proof of the claim. ■

3.5 Robustness

In this section, we study a natural scenario for extending the considered setup. In particular, it is assumed that a noisy channel is present between the legitimate parties and the goal is to study the robustness of the framework and the proposed solution when channel noise is present.

The revised system model, shown in Figure 3.2, is as follows: the channel between Alice and Bob is no longer noiseless, and it can be a certain type of channel to be studied, e.g., binary

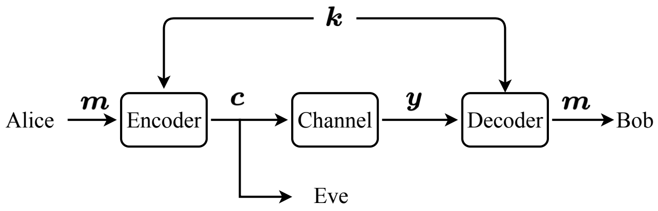


Figure 3.2: Modified setup for the proposed coding scheme in the presence of a noisy channel.

symmetric channel (BSC), binary erasure channel (BEC), additive-white Gaussian noise channel (AWGN), etc. However, for the eavesdropper, we still consider a worst-case scenario from the legitimate parties' perspective. In other words, it is assumed that Eve receives the transmitted codeword through a noiseless channel, and hence, she has access to the codeword error-free. Alice aims to utilize a coding scheme such that the threshold security requirement at Eve is satisfied while establishing a reliable communication with Bob that is robust in the presence of channel noise.

Note that the assumption on Eve's observation here makes it reasonable to keep the conditions in (3.2.2) and (3.2.3) the same in this revised model. On the other hand, the reliability condition in (3.2.1) needs to be modified to account for the noisy channel. We do this from a conventional block coding perspective where reliability is measured in terms of a certain number of errors and erasures that can be corrected. More specifically, the reliability condition is still stated as

$$H(\boldsymbol{m}|\boldsymbol{y},\boldsymbol{k}) = 0, \tag{3.5.1}$$

provided that the number of erasures and errors introduced in $\boldsymbol{y}$ satisfies a certain condition that depends on the underlying coding scheme. For instance, consider coding schemes based on linear block codes. Suppose that the minimum distance of the robustness coding scheme is $D_{\min}$ when the key is fixed, which is different from the minimum distance of the threshold security coding scheme, i.e., $d_{\min}$. Then the condition on the number of errors and erasures is simply $2\tau_e + \rho_e \leqslant D_{\min} - 1$, where τ_e is the number of errors and ρ_e is the number of erasures, same as in conventional block codes.

In the remainder of this section, we discuss a general method to construct codes for threshold security and robustness as well as describe an explicit low-complexity construction based on Reed-Muller codes for binary erasure channels along with an SC decoder.

3.5.1 General construction

A straightforward solution to construct coding schemes for the setting described in this section is by utilizing the concatenation of two codes. More specifically, a coding scheme, constructed to guarantee the desired threshold security in the error-free case, would be concatenated with an inner code, that can be an off-the-shelf block code, to guarantee the desired reliability for the Alice and Bob communication. Although this solution is straightforward, one needs to ensure that the threshold security guarantee is not compromised when more redundancy is added through the inner encoder which will be then revealed to Eve.

In the aforementioned concatenation scheme, the overall encoder and decoder at Alice and Bob, respectively, are referred to as *supercoder* and *superdecoder*, respectively. The construction of the concatenated scheme is described in more detail next. Consider a *proper* coding scheme, that guarantees threshold security requirement, that is obtained from an $[n, m, d_{\min}]_q$ linear block code with the generator matrix $\boldsymbol{H}^T$. Also, consider an error-correcting code, used as an inner code to guarantee the reliability, that is an $[N, m, D_{\min}]_q$ linear block code with the generator matrix denoted by $\boldsymbol{G}_r$. It is important to note that both codes have the same dimension m.

The encoding process is as follows. First, $\boldsymbol{u} = \pi(\boldsymbol{k}, \boldsymbol{m})$ is passed through the outer threshold security encoder that multiplies $\boldsymbol{u}$ by $\boldsymbol{H}$. The result is then passed to the inner encoder, which multiplies its input by $\boldsymbol{G}_r$. Then the resulting codeword $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{H}\boldsymbol{G}_r$ is transmitted to Bob through the noisy channel. Bob receives a corrupted version of the codeword $\boldsymbol{c}$, denoted as $\boldsymbol{y}$, and passes it through the decoder consisting of an inner decoder and an outer decoder. The inner decoder retrieves $\boldsymbol{c}' = \boldsymbol{u}\boldsymbol{H}$. Note that we have $\boldsymbol{c}'$ error-free provided that the number of errors and/or erasures satisfies the given condition on the reliability guarantee of the inner code. Then $\boldsymbol{c}'$ together with the key $\boldsymbol{k}$ are passed through the outer decoder, designed for the threshold security coding scheme; hence, retrieving $\boldsymbol{m}$. The following lemma states that this construction does not compromise the key and threshold security conditions.

Lemma 3.5.1 *The aforementioned concatenation coding scheme results in a t-threshold secure code.*

Proof: To show that the lemma holds, we need to have $\operatorname{rank}(\boldsymbol{H}\boldsymbol{G}_r) = m$, $\operatorname{rank}(\boldsymbol{H}_{\mathcal{A}}\boldsymbol{G}_r) = m$, $\operatorname{rank}(\boldsymbol{H}_{\mathcal{A}^c}\boldsymbol{G}_r) = k$, and $\operatorname{rank}(\boldsymbol{H}_{\mathcal{B}^c}\boldsymbol{G}_r) = m$, where $\mathcal{A}$ and $\mathcal{A}^c$ are chosen such that the code is *proper*, as stated in Definition 3.2, and $\mathcal{B}^c$ is as defined in Theorem 3.3.4. It can be observed that all these equations hold simply because $\boldsymbol{G}_r$ is full row rank. ■

3.5.2 Low-complexity construction

In this section, we aim at presenting a unified coding scheme, for threshold security and robustness, that can be decoded using one unified SC decoder. This would potentially result in more efficient hardware implementation and improved latency compared to the general concatenated scheme.

In particular, a scenario with binary symbol erasures is considered, where at most $\rho_e = D_{\min} - 1$ erasures are assumed to occur with $D_{\min}$ being the minimum distance of the underlying code. For the proposed coding scheme, an encoder is presented together with a superdecoder that simultaneously corrects erasures and decodes the message using the key. To this end, the coding scheme presented for noiseless channels in Section 3.4 is extended to be utilized along with an RM-based code to handle binary erasures.

3.5.2.1 Encoder

In the considered scheme, the same RM code is used for threshold security and robustness. More specifically, an RM(s, r) is used, which is as previously described, a $[2^s, \sum_{i=0}^{r} \binom{s}{i}, 2^{s-r}]_2$ with the generator matrix denoted by $\boldsymbol{G}(s, r)$. The encoder with input $\boldsymbol{u}$, consisting of both the message and the key, outputs the codeword $\boldsymbol{c}$ specified as follows:

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}^T(s, r)\boldsymbol{G}(s, r) = \boldsymbol{u}\widetilde{\boldsymbol{G}}(s, r), \tag{3.5.2}$$

where $\widetilde{\boldsymbol{G}}(s, r)$ is a notation introduced here to denote $\boldsymbol{G}^T(s, r)\boldsymbol{G}(s, r)$. Note that the encoder can be implemented recursively, since $\widetilde{\boldsymbol{G}}(s, r)$ can be expressed recursively as follows,

$$\widetilde{\boldsymbol{G}}(s, r) = \boldsymbol{G}^T(s, r)\boldsymbol{G}(s, r), \tag{3.5.3}$$

$$\widetilde{\boldsymbol{G}}(s,r) = \begin{bmatrix} \boldsymbol{G}^T(s-1,r-1) & \boldsymbol{G}^T(s-1,r) \\ \boldsymbol{0} & \boldsymbol{G}^T(s-1,r) \end{bmatrix} \begin{bmatrix} \boldsymbol{G}(s-1,r-1) & \boldsymbol{0} \\ \boldsymbol{G}(s-1,r) & \boldsymbol{G}(s-1,r) \end{bmatrix}, \tag{3.5.4}$$

$$= \begin{bmatrix} \boldsymbol{G}^T(s-1,r-1)\boldsymbol{G}(s-1,r-1) + \boldsymbol{G}^T(s-1,r)\boldsymbol{G}(s-1,r) & \boldsymbol{G}^T(s-1,r)\boldsymbol{G}(s-1,r) \\ \boldsymbol{G}^T(s-1,r)\boldsymbol{G}(s-1,r) & \boldsymbol{G}^T(s-1,r)\boldsymbol{G}(s-1,r) \end{bmatrix}, \tag{3.5.5}$$

$$= \begin{bmatrix} \widetilde{\boldsymbol{G}}(s-1,r-1) + \widetilde{\boldsymbol{G}}(s-1,r) & \widetilde{\boldsymbol{G}}(s-1,r) \\ \widetilde{\boldsymbol{G}}(s-1,r) & \widetilde{\boldsymbol{G}}(s-1,r) \end{bmatrix}. \tag{3.5.6}$$

Note that the encoder described by (3.5.2) utilizes the construction presented in Section 3.4.1, which achieves threshold security parameter $t = 2^{s-r} - 1$, and we use the same choice of indices dedicated for the key and the message that results in a *proper* code.

3.5.2.2 Decoder

We present a unified SC superdecoder for the coding scheme described above that corrects $\rho_e \leqslant D_{\min} - 1$ erasures, where $D_{\min} = 2^{s-r}$, and recovers the message given the shared key. The recursive decoder takes the received bit sequence y_1^n, the shared key $\boldsymbol{k}$, key indices $\mathcal{A}^c$, code parameters s, r, and a recursion parameter j as inputs. Initially, $j = 1$. It outputs b_1^n, which is equal to the codeword $\boldsymbol{c}$ provided that $\rho_e \leqslant D_{\min} - 1$, as well as $u_1^n = \pi(\boldsymbol{k}, \boldsymbol{m})$, which is used to retrieve the message $\boldsymbol{m}$, and a recursion index j' used to track the index of the last decoded bit. Pseudocode for the decoder is shown in Algorithm 3.2. The following claim shows the success of the described decoder.

Algorithm 3.2 Unified SC decoder for binary erasures (DecBE)

```
Input: $\boldsymbol{k}$, $y_1^n$, $\mathcal{A}^c$, $s$, $r$, $j$.
Output: $b_1^n$, $u_1^n$, $j'$.
if $r = 0$ then
    $\mathcal{I} = [j, j+1, ...., j + 2^s - 1]$
    $i_1 \leftarrow$ index of any non-erasure bit in $y_1^n$.
    for $i \in \mathcal{A}^c$ do
        $u_i = k_i$
    end for
    $i' \in \mathcal{I} \setminus \mathcal{A}^c$
    $u_{i'} = y_{i_1} \oplus_{i \in \mathcal{A}^c} u_i$
    $b_1^n = [y_{i_1}, y_{i_1}, ..., y_{i_1}]$
    $j' = j + 2^s - 1$
else
    $\bar{\boldsymbol{y}} = y_1^{n/2} \oplus y_{n/2+1}^n$
    $\boldsymbol{b}'_1, u_1^{n/2}, j'_1 \leftarrow$ DecBE($\boldsymbol{k}_1, \bar{\boldsymbol{y}}, \mathcal{A}_1^c, s-1, r-1, j$)
    $\boldsymbol{b}'_2 = u_1^{n/2} \widetilde{\boldsymbol{G}}(s-1, r)$
    $\boldsymbol{b}' = [\boldsymbol{b}'_1 \oplus \boldsymbol{b}'_2, \boldsymbol{b}'_2]$
    $\tilde{y}_1^n = y_1^n \oplus \boldsymbol{b}' = [\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{y}}_2]$
    $l = \underset{j \in 1,2}{\text{argmin}}$ (number of erasures in $\tilde{\boldsymbol{y}}_j$)
    $\boldsymbol{b}''_1, u_{n/2+1}^n, j' \leftarrow$ DecBE($\boldsymbol{k}_2, \tilde{\boldsymbol{y}}_l, \mathcal{A}_2^c, s-1, r, j'_1 + 1$)
    $\boldsymbol{b}'' = [\boldsymbol{b}''_1, \boldsymbol{b}''_1]$
    $b_1^n = \boldsymbol{b}' \oplus \boldsymbol{b}''$
end if
return $u_1^n$, $b_1^n$, $j'$
```

Claim 3.5.2 *The proposed unified RM-based coding scheme together with the unified SC superdecoder in Algorithm 3.2 successfully retrieves the message as long as* $\rho_e \leqslant D_{\min} - 1$.

Proof: Let the received sequence be denoted by y_1^n which has at most ρ_e erasures. Let also the key bits be denoted by $\boldsymbol{k}$ which are assigned to entries of $\boldsymbol{u}$ indexed by elements of $\mathcal{A}^c$. We use induction on the parameter s of the underlying RM code of length 2^s to prove the claim. The induction hypothesis is that the decoder is successful for any RM-based coding scheme of length 2^s with some parameter $r \leqslant s$, and a key with size $\sum_{i=r+1}^{s} \binom{s}{i}$, assuming there are at most $\rho_e = 2^{s-r} - 1$ erasures. The induction base is $s = 0$, for which the induction hypothesis is trivial.

Now, suppose that the induction hypothesis holds for s and we want to show it for $s+1$.

Case 1: $r = 0$, i.e., we have an RM$(s+1, 0)$ which becomes a repetition code of length $n = 2^{s+1}$. In this case, $\widetilde{\boldsymbol{G}}(s+1, 0)$ is the all-ones matrix and the entries of codeword are all equal to the sum of entries in $\boldsymbol{u}$. Note that the number of message bits is $m = \sum_{i=0}^{r} \binom{s+1}{i} = 1$ and we have $2^{s+1} - 1$ key bits. Also, the maximum number of erasures the code can correct is $2^{s+1} - 1$. Hence, the decoder successfully retrieves the message bit using the non-erasure symbols, of which there is at least one, in y_1^n. Suppose that the non-erasure bit is indexed by i_1. Since the locations of the key bits are known, we can place them at their respective locations retrieving u_i's for all $i \in \mathcal{A}^c$. Next, the message bit located at i' is retrieved as $u_{i'} = y_{i_1} \oplus_{i \in \mathcal{A}^c} u_i$, and the corresponding codeword is also retrieved correctly. Hence, the decoder is successful. Note that this case corresponds to lines 4-12 of Algorithm 3.2.

Case 2: $r > 0$. The code length is $n = 2^{s+1}$ and the key length is $\sum_{i=r+1}^{s+1} \binom{s+1}{i}$. We split the key indices into two parts, namely $\mathcal{A}_1^c$ and $\mathcal{A}_2^c$, representing the key bits $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ in the first and the second half sub-blocks of $\boldsymbol{u}$, respectively. The lengths of $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ are $|\mathcal{A}_1^c| = \sum_{i=r}^{s} \binom{s}{i}$ and $|\mathcal{A}_2^c| = \sum_{i=r+1}^{s} \binom{s}{i}$, respectively, due to the aforementioned choice of indices. The decoder first computes $\bar{\boldsymbol{y}} = y_1^{n/2} \oplus y_{n/2+1}^{n}$ which will have at most $2^{s+1-r} - 1$ erasures. It then passes this to the left child, in the binary tree representation terminology discussed earlier, along with $\boldsymbol{k}_1$ and the set of its corresponding indices $\mathcal{A}_1^c$. The left child decodes a codeword of length $n' = 2^s$ using a code with parameter $r' = r - 1 \geqslant 0$, which can correct up to $2^{s-r'} - 1 = 2^{s+1-r} - 1$ erasures, and retrieves the message bits in $u_1^{n/2}$ given the key $\boldsymbol{k}_1$ of length $\sum_{i=r'+1}^{s} \binom{s}{i} = \sum_{i=r}^{s} \binom{s}{i}$. The decoder on the left child is successful by induction hypothesis. It outputs $u_1^{n/2}$ and $\boldsymbol{b}_1'$. After that, the decoder computes $\boldsymbol{b}_2' = u_1^{n/2} \widetilde{\boldsymbol{G}}(s, r)$ followed by $\boldsymbol{b}' = [\boldsymbol{b}_1' \oplus \boldsymbol{b}_2', \boldsymbol{b}_2']$. Then, the decoder computes

$\tilde{y}_1^n = y_1^n \oplus \boldsymbol{b}'$ and chooses either $\tilde{y}_1^{n/2}$ or $\tilde{y}_{n/2+1}^n$, whichever has a smaller number of erasures, and passes it to the right child together with $\boldsymbol{k}_2$ and the corresponding set of indices $\mathcal{A}_2^c$. The number of erasures in what is passed to this child is at most $2^{s+1-r}/2 - 1 = 2^{s-r} - 1$, and the length of the key is $\sum_{i=r+1}^{s} \binom{s}{i}$. The decoder on this child decodes a codeword of length $n' = 2^s$ using a code with parameter $r' = r > 0$, which can correct up to $2^{s-r'} - 1 = 2^{s-r} - 1$ erasures and retrieves the message bits in $u_{n/2+1}^n$ using the key $\boldsymbol{k}_2$ of length $\sum_{i=r'+1}^{s} \binom{s}{i} = \sum_{i=r+1}^{s} \binom{s}{i}$. Decoding here is also successful by induction hypothesis. It outputs $u_{n/2+1}^n$ and $\boldsymbol{b}_1''$. The overall decoder then computes $\boldsymbol{b}'' = [\boldsymbol{b}_1'', \boldsymbol{b}_1'']$ and outputs $b_1^n = \boldsymbol{b}' \oplus \boldsymbol{b}''$ and u_1^n. Hence, u_1^n is retrieved and the proof is complete. ■

3.6 Conclusion

This chapter formulated the problem of threshold-secure coding with a shared key based on information-theoretic measures. The problem of threshold-secure coding includes a threshold parameter that is to be designed based on the application for such coding schemes. A threshold-secure coding scheme based on error-correcting linear block codes is presented where the parameter t of the threshold-secure scheme is shown to be directly related to the minimum distance of the underlying linear block code. Furthermore, a coding scheme based on Reed-Muller codes is described. Its encoder is computed recursively and is shown to satisfy the conditions for a *proper* code, and its decoder is a successive cancellation decoder and its success is shown. Moreover, an extended setup taking into account the noise in the communication channel between legitimate parties is considered. Then, a robust and threshold-secure coding scheme, based on code concatenation, is presented for general channels. Also, a unified coding scheme built upon Reed-Muller codes for both threshold security and robustness in the presence of erasures is described, and its successive cancellation decoder is also presented and its success is shown.

CHAPTER 4

Coded Machine Unlearning

4.1 Introduction

This chapter presents the coded machine unlearning algorithm. Given the abundance of data, machine learning has become ubiquitous in the past decade. Once an ML model is trained, some samples in the training dataset might be required to be *unlearned* due to various reasons, e.g., to satisfy users' requests of data removal, or due to discovery of corrupt low-quality samples or adversarially modified samples that are specifically created to adversely affect the performance of the ML model. As ML models may be arbitrarily complex and may be trained on large datasets, it is important to devise unlearning methods that are efficient. The problem of efficiently removing information about a training sample from a trained ML model, referred to as *machine unlearning* [29], was recently introduced in the literature. In this scenario, a trustworthy party aims to train an ML model on a training dataset of raw data with the guarantee that unlearning requests are satisfied by removing the sample from the training dataset as well as removing any trace of them in the trained ML model. One straightforward approach to perfectly satisfy this requirement is to retrain the model from scratch after removing the samples that need to be unlearned from the training dataset. However, as large training datasets are increasingly available and used in these

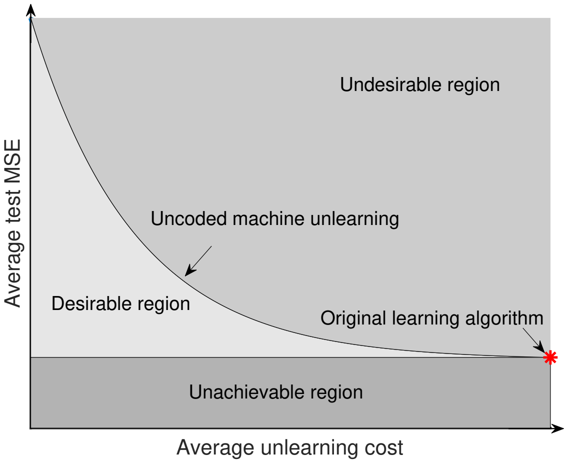


Figure 4.1: Sketch of performance vs unlearning cost trade-off of the baseline uncoded machine unlearning algorithm proposed in [2].

models, retraining after receiving each unlearning request becomes prohibitively expensive.

This chapter presents an efficient algorithm for perfect machine unlearning in regression problems. The proposed algorithm builds upon a machine unlearning algorithm that uses ensemble learning methods to efficiently remove samples from ML models [2]. The baseline algorithm provides a general framework for perfect unlearning that considers an ensemble learning setup where a master node shards the training dataset and assigns the shards to non-communicating weak learners that are trained independently from each other, and then aggregates their models using a certain aggregation function. The proposed algorithm in this chapter is designed for regression problems where the server utilizes random coding to encode shards into compressed shards that are then used to train weak learners. Experimental results are conducted to evaluate the proposed algorithm against the baseline in terms of performance vs unlearning cost trade-off, which show the superior performance of the proposed algorithm compared to the baseline.

4.2 Problem Setup

Consider a setup where the training dataset $\boldsymbol{D}$ is a matrix denoted as $[\boldsymbol{X}, \boldsymbol{y}]$ whose rows are the independent and identically distributed samples $\boldsymbol{x}_i$ along with their responses y_i for $i = 1, 2, ..., n_t$, where $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Denote n_t as the total number of samples in the dataset, and d as the number of features of each sample. Columns of $\boldsymbol{X}$ are referred to as the features while $\boldsymbol{y}$ is referred to as the response variable, whose elements are of the form

$$y_i = f(\boldsymbol{x}_i) + n_i, \tag{4.2.1}$$

where n_i is the Gaussian noise term. The training dataset is used to train a learning model to produce a model, i.e., a function $f : \mathcal{X} \to \mathbb{R}$, that minimizes a loss function. For regression problems, the loss function $\ell(\cdot)$ is a function that measures the goodness of fit of the model $f \in \mathcal{F}$ on the training dataset, typically expressed as

$$\ell(\boldsymbol{X}, \boldsymbol{y}; f) = \frac{1}{n_t} \sum_{i=1}^{n_t} (y_i - f(\boldsymbol{x}_i))^2 + \Omega(\|f\|_{\mathcal{F}}), \tag{4.2.2}$$

where $\Omega(\cdot)$ is a regularization term. The learning model finds a function f^* that minimizes the loss function as follows

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \ell(\boldsymbol{X}, \boldsymbol{y}; f). \tag{4.2.3}$$

The Representer theorem is a powerful theorem for general regression problems. It states that for the regularized loss in (4.2.2), when using a a strictly increasing function Ω, and a kernel function

$f_k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $\mathcal{F}$ as its associated Reproducing Kernel Hilbert Space (RKHS), then the minimizer f^* of the loss function above is expressed in the form [70]

$$f^*(\cdot) = \sum_{i=1}^{n_t} w_i f_k(\cdot, \boldsymbol{x}_i), \tag{4.2.4}$$

where $w_i \in \mathbb{R}$ is to be estimated. This powerful theorem translates any regression problem, even nonlinear problems, as a linear problem in the RKHS. Hence, the problem can be transformed and re-expressed to be as follows

$$\boldsymbol{y} = \boldsymbol{F}_k \boldsymbol{w} + \boldsymbol{n}, \tag{4.2.5}$$

where $\boldsymbol{F}_k$ is an $n_t \times n_t$ kernel matrix whose elements $\boldsymbol{F}_k(i, j) = f_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{w}$ is a $n_t \times 1$ coefficient vector, and $\boldsymbol{n}$ is a $n_t \times 1$ noise vector. The L_2-regularized learning model for this kernel problem, referred to as ridge regression, aims to estimate $\boldsymbol{w}$ that minimizes the loss function

$$\ell(\boldsymbol{F}_k, \boldsymbol{y}; \boldsymbol{w}) = \frac{1}{n_t} \sum_{i=1}^{n_t} (y_i - \boldsymbol{f}_{k,i}^T \boldsymbol{w})^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}, \tag{4.2.6}$$

where $\boldsymbol{f}_{k,i}$ is the i-th row of $\boldsymbol{F}_k$, and λ is the regularization parameter. We denote the resulting model trained on $[\boldsymbol{X}, \boldsymbol{y}]$ when initialized with parameters $\boldsymbol{\theta}$ as $f^* = \mathcal{M}^{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y})$.

These kernel methods suffer greatly in regimes where the size of training datasets is large. Specifically, for a dataset with a fixed number of features, computations of the elements in the kernel matrix result in an additional complexity of $O(n_t^2)$ on top of the optimization method used to solve the problem. One method of resolving this issue is proposed in [71], which suggests using random projections of the features to a relatively low-dimensional space compared to n_t. This gives a good approximation of the function f^* using random projections to a D-dimensional space where $d < D \ll n_t$, which enables efficient linear regression methods to be used to solve the regression problem. These random projections enable an approximation of the target function f^*, denoted as $\hat{f}$, expressed as follows

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{D} \Theta(\boldsymbol{x}^T \boldsymbol{\Phi}_i + \varsigma_i) w_i + n, \tag{4.2.7}$$

where $\Theta(\cdot)$ is an activation function, $\boldsymbol{\Phi}_i$ and ς_i are chosen randomly from some distributions, w_i's are the coefficients to be estimated, and D is the desired dimension of the projected features [71]. This enables us to apply this transformation of the original feature matrix $\boldsymbol{X}$ into another feature matrix $\boldsymbol{X}_p$ of size $n_t \times D$, then we have the following

$$\boldsymbol{y} = \boldsymbol{X}_p \boldsymbol{w} + \boldsymbol{n}, \tag{4.2.8}$$

where $\boldsymbol{w}$ is the coefficient vector to be estimated of size $D \times 1$.

After the model has been trained, it is used for prediction until a request of unlearning samples arrives. Once unlearning requests arrive, the model stops processing any prediction requests and launches the unlearning algorithm. Machine unlearning is formulated as follows: when an unlearning request of sample $[\boldsymbol{x}_u^T, y_u]$ from the training dataset is received, the model must be immediately updated to remove any effect of this sample, i.e., unlearn it, from $\mathcal{M}^{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y})$. The unlearning algorithm is denoted as $\mathcal{U}$ and its output is an updated model denoted as $\mathcal{U}(\mathcal{M}^{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y}), [\boldsymbol{x}_u^T, y_u])$. We require $\mathcal{U}$ to be a perfect unlearning algorithm defined as follows [2].

Definition 4.1 (perfect machine unlearning) *An unlearning algorithm* $\mathcal{U}$ *on model* $\mathcal{M}^{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y})$ *is* *said to be* perfect *if the output of the unlearning algorithm removing the sample* $[\boldsymbol{x}_u^T, y_u]$*, denoted as* $\mathcal{U}(\mathcal{M}^{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y}), [\boldsymbol{x}_u^T, y_u])$*, is a statistical draw from the distribution of the models trained on* $[\boldsymbol{X} \setminus \boldsymbol{x}_u, \boldsymbol{y} \setminus y_u]$*, denoted as* $\mathcal{M}^{\boldsymbol{\theta}}(\boldsymbol{X} \setminus \boldsymbol{x}_u, \boldsymbol{y} \setminus y_u)$*, where* $[\boldsymbol{X}\setminus\boldsymbol{x}_u, \boldsymbol{y}\setminus y_u]$ *denotes the training dataset* $[\boldsymbol{X}, \boldsymbol{y}]$ *after removing the sample* $[\boldsymbol{x}_u^T, y_u]$ *from it.*

Perfect unlearning algorithms ensure the complete removal of samples from the model but may suffer in terms of their efficiency. Removing the samples from the training dataset and retraining a model from scratch achieves perfect unlearning. However, the major hurdle of this approach is the extended delay time required to unlearn a sample as retraining is the process that mainly causes this delay; hence, it is desirable to design efficient unlearning algorithms that can be used for large scale datasets.

4.3 Proposed Algorithm

The proposed algorithm is described in two parts, learning and unlearning. In the learning phase, we present a method for encoding the training dataset prior to training and describe a specific coding scheme for a regression learning model. After the model has been trained, we transition into the unlearning phase, we describe an efficient method to process unlearning requests using the coded training dataset and update the model to perfectly unlearn the desired samples.

4.3.1 Learning

The proposed algorithm introduces the idea of data encoding prior to training the ensemble model as shown in Figure 4.2. The main learner $\mathfrak{M}$, also referred to as the master node, is launched to learn a regression model whose training dataset is assumed to have been preprocessed. The model

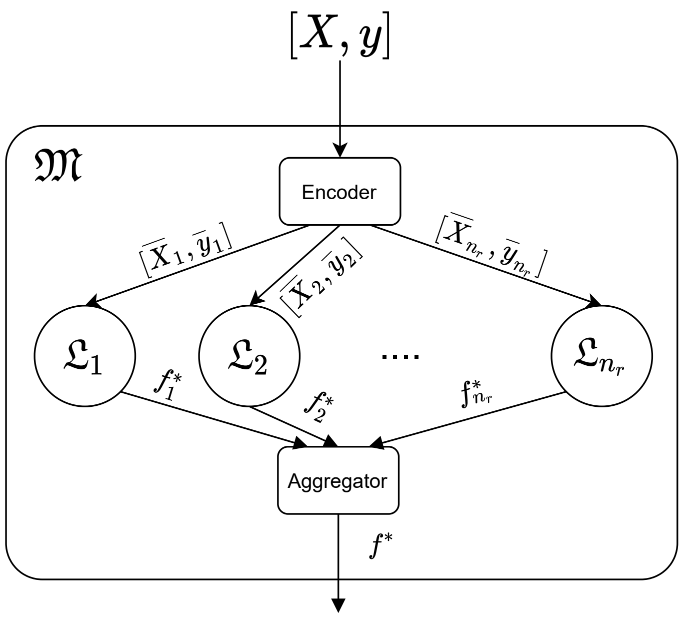


Figure 4.2: Proposed coded learning setup.

starts by passing the training dataset through an *encoder* to produce a sharded coded training dataset that contains n_r coded shards. Then, each coded shard j is used to train a weak learner $\mathfrak{L}_j$ to produce a model denoted as f_j^*. Once these weak learners are trained, the model is ready for prediction. When sample $\boldsymbol{x}$ is passed to $\mathfrak{M}$, it is directly passed to each of the weak learners to produce weak predictions $f_j^*(\boldsymbol{x})$ for $j = 1, 2, ..., n_r$, then $\mathfrak{M}$ computes the final prediction $f^*(\boldsymbol{x})$ by applying an aggregation function $a : \mathbb{R}^{n_r} \to \mathbb{R}$, such as averaging, a majority vote, etc, as follows

$$f^*(\boldsymbol{x}) = a(f_1^*(\boldsymbol{x}), f_2^*(\boldsymbol{x}), ..., f_{n_r}^*(\boldsymbol{x})). \tag{4.3.1}$$

For linear regression models, or nonlinear regression models coupled with random projections, we know that the generated model f_j^* is the corresponding weights $\boldsymbol{w}_j^*$. Once all weak learners $\mathfrak{L}_j$'s have been trained, $\mathfrak{M}$ produces a matrix $\boldsymbol{W}^*$ whose columns are the estimated coefficients from the weak learners as follows

$$\boldsymbol{W}^* = [\boldsymbol{w}_1^*, \boldsymbol{w}_2^*, ..., \boldsymbol{w}_{n_r}^*]. \tag{4.3.2}$$

Once $\boldsymbol{W}^*$ is available, the model $\mathfrak{M}$ computes the aggregate prediction weights by computing the mean of the weight vectors of the weak learners to produce $\boldsymbol{w}_{\text{agg}}^*$, which is directly used at the time of prediction, bypassing the weak predictions computations. When the sample $\boldsymbol{x}$ is passed as input, the predicted output is

$$f^*(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w}_{\text{agg}}^*. \tag{4.3.3}$$

The encoding of the training dataset is a method to produce a new training dataset with the goal of reducing learning and unlearning costs. Coding can be viewed as a method to incorporate multiple samples from the uncoded training dataset into a single sample of the coded training dataset to enable efficient learning and unlearning. In other words, although we have fewer coded samples for weak learners, each of these coded samples is created from multiple uncoded samples, enabling the model to learn these uncoded samples indirectly. First, let us define an encoder as follows.

Definition 4.2 (encoder) *An encoder with rate R_c is defined as a function that transforms the original training dataset with n_t samples into another dataset with n_c samples while maintaining the same number of features. The rate of this encoder is*

$$R_c = \frac{n_t}{n_c}. \tag{4.3.4}$$

Algorithm 4.1 Learning (Learn)

```
Input: $[\boldsymbol{X}, \boldsymbol{y}], n_s, n_r, \rho_G$.
Output: $\boldsymbol{W}^*, \{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}, \boldsymbol{G}$.
At master node $\mathfrak{M}$, do
if $n_s \neq 1$ then
    $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}, \boldsymbol{G} \leftarrow \text{LinearEnc}([\boldsymbol{X}, \boldsymbol{y}], n_s, n_r, \rho_G)$.
else
    $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\} = \{[\boldsymbol{X}, \boldsymbol{y}]\}$
    $\boldsymbol{G} = [1]$
end if
Send $[\overline{\boldsymbol{X}}_i, \overline{\boldsymbol{y}}_i]$ to weak learner $\mathfrak{L}_i$
At weak learner $\mathfrak{L}_i$, do
$\boldsymbol{w}_i^* \leftarrow \text{argmin}_{\boldsymbol{w}}\, \ell([\overline{\boldsymbol{X}}_i, \overline{\boldsymbol{y}}_i], \boldsymbol{w})$
Send $\boldsymbol{w}_i^*$ to $\mathfrak{M}$
At master node $\mathfrak{M}$, do
$\boldsymbol{W}^* = [\boldsymbol{w}_1^*, \boldsymbol{w}_2^*, ..., \boldsymbol{w}_{n_r}^*]$
$\boldsymbol{w}_{\text{agg}}^* = \frac{1}{n_r} \sum_{i=1}^{n_r} \boldsymbol{w}_i^*$
```

When using shards of equal size, as considered in this work, the rate can also be viewed as the ratio of the number of uncoded shards to the number of coded shards. The rate and design of the encoder are now additional parameters of the model that require consideration when training a model. It is worth noting that the design of the encoder itself for a fixed rate directly affects the unlearning cost of the overall model as well as the performance, as will be clarified later. Hence, it should be carefully considered when designing a model.

The proposed coded learning model for linear regression is described in Algorithm 4.1. The learning algorithm takes the training dataset $[\boldsymbol{X}, \boldsymbol{y}]$ and code parameters n_s, n_r, ρ_G as inputs and outputs the coded training dataset along with the coding matrix and the trained model. The node $\mathfrak{M}$ utilizes a linear encoder described in Algorithm 4.2 to encode the training dataset using the provided code parameters. The linear encoder takes the desired code parameters n_s, n_r, ρ_G along with the training dataset $[\boldsymbol{X}, \boldsymbol{y}]$ as inputs and processes it as follows: $[\boldsymbol{X}, \boldsymbol{y}]$ is divided into n_s disjoint submatrices of equal size, i.e., each has $\overline{n}_t = \frac{n_t}{n_s}$ samples, denoted as $[\boldsymbol{X}_i, \boldsymbol{y}_i]$ for $i =$

Algorithm 4.2 Linear encoder (LinearEnc)

```
Input: $[\boldsymbol{X}, \boldsymbol{y}]$, $n_s$, $n_r$, $\rho_G$.
Initialization: $\boldsymbol{G} = \boldsymbol{0}_{n_s \times n_r}$, $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}$ = empty.
Output: $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}$, $\boldsymbol{G}$.
while $\boldsymbol{G}$ is not full column rank do
    Set $\boldsymbol{G} = \boldsymbol{0}_{n_s \times n_r}$
    while $\boldsymbol{G}$ has any all-zero rows do
        $\boldsymbol{G} \leftarrow \mathrm{RandMatrix}(n_s, n_r, \rho_G)$
    end while
end while
Split $[\boldsymbol{X}, \boldsymbol{y}]$ into $n_s$ submatrices $[\boldsymbol{X}_i, \boldsymbol{y}_i]$ of equal size
for $j$ in $\mathrm{range}(n_r)$ do
    $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}.\mathrm{append}([(\sum_{i=1}^{n_s} g_{ij}\boldsymbol{X}_i), (\sum_{i=1}^{n_s} g_{ij}\boldsymbol{y}_i)])$
end for
return $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}, \boldsymbol{G}$
```

$1, 2, ..., n_s$. These are encoded using a matrix $\boldsymbol{G}$, described next, to produce $[\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{y}}_j]$, for $j = 1, 2, ..., n_r$. The output of this encoder is $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}$ whose elements are the coded shards used to train the corresponding weak learners, i.e., shard j is used to train the j-th weak learner to produce the corresponding optimum weights $\boldsymbol{w}_j^*$. Note that the code parameters should keep the coded shards in the original regime of the original dataset; for example, if $n_t > d$, then $\overline{n}_t > d$.

Random codes have been useful in information theory [72] and random projections in signal processing [73, 74] and machine learning [71]; hence, we propose to use a random binary matrix generator (RandMatrix) to generate $\boldsymbol{G}$. In this algorithm, the matrix $\boldsymbol{G}$ is of size $n_s \times n_r$ and density $0 < \rho_G \leqslant 1$. We desire the matrix $\boldsymbol{G}$ to be a tall matrix, i.e., $n_r \leqslant n_s$, since our goal is to reduce the number of coded samples used for training; otherwise there will be redundancy of samples in multiple shards leading to more operations needed to remove a sample. Since $n_r \leqslant n_s$, $\boldsymbol{G}$ needs to satisfy two conditions: each row of $\boldsymbol{G}$ should have at least one nonzero element, and it should have full rank. The first condition ensures that all the shards are used in training the model, while the second condition ensures that every weak learner has a training dataset that is unique from all

Algorithm 4.3 Unlearning (Unlearn)

```
Input: $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}, [\boldsymbol{X_i}, \boldsymbol{y_i}], \boldsymbol{i}, \boldsymbol{G}, \boldsymbol{W}^*$.
Initialization: $\boldsymbol{j} =$ empty.
Output: $\{\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{y}}\}, \widetilde{\boldsymbol{W}}^*$.
**At master node $\mathfrak{M}$, do**
Set $\{\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{y}}\} = \{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}$
Set $\widetilde{\boldsymbol{W}}^* = \boldsymbol{W}^*$
**for** $l$ in $\mathrm{range}(\mathrm{length}(\boldsymbol{i}))$ **do**
    $s' \leftarrow$ index of the uncoded shard containing $[\boldsymbol{x}_l^T, y_l]$
    $l' \leftarrow$ index of $[\boldsymbol{x}_l^T, y_l]$ within shard $s'$
    $\boldsymbol{j}' \leftarrow$ indices of nonzero elements in row $s'$ of $\boldsymbol{G}$
    **for** $j'$ in $\boldsymbol{j}'$ **do**
        $\widetilde{\boldsymbol{x}}_{j'i'} = \widetilde{\boldsymbol{x}}_{j'i'} - g_{s'j'}\boldsymbol{x}_i$
        $\widetilde{y}_{j'i'} = \widetilde{y}_{j'i'} - g_{s'j'}y_i$
    **end for**
    $\boldsymbol{j}.\mathrm{append}(\boldsymbol{j}')$
**end for**
$\boldsymbol{j}_u \leftarrow \mathrm{unique}(\boldsymbol{j})$
Send $[\widetilde{\boldsymbol{X}}_j, \widetilde{\boldsymbol{y}}_j]$ to weak learner $\mathfrak{L}_j$ for all $j \in \boldsymbol{j}_u$
**At weak learner $\mathfrak{L}_j$, do**
$\tilde{\boldsymbol{w}}_j^* \leftarrow \mathrm{argmin}_{\boldsymbol{w}}\, \ell(\widetilde{\boldsymbol{X}}_j, \widetilde{\boldsymbol{y}}_j; \boldsymbol{w})$
Discard the previous model $\boldsymbol{w}_j^*$
Send $\tilde{\boldsymbol{w}}_j^*$ to $\mathfrak{M}$
**At master node $\mathfrak{M}$, do**
Replace column $j$ of $\widetilde{\boldsymbol{W}}^*$ with the updated $\tilde{\boldsymbol{w}}_j^*$ for all $j \in \boldsymbol{j}_u$
Set $\tilde{\boldsymbol{w}}^*_{\text{agg}} = \frac{1}{n_r}\sum_{i=1}^{n_r} \tilde{\boldsymbol{w}}_i^*$
```

other weak learners. Another consequence of using a code with $n_r \leqslant n_s$ is that it lowers the initial learning cost by a factor of R_c compared to uncoded machine unlearning. For example, for some n_s and n_r then we only need to train n_r learners each using n_t/n_s coded samples, compared to n_s learners each with n_t/n_s uncoded samples in uncoded machine unlearning.

4.3.2 Unlearning

Now that the model has been trained on the coded training dataset, we proceed to describe an algorithm to unlearn samples from this model. Our goal is to remove such samples from the coded shards as well as to remove any trace of such samples from the affected weak learners where such samples appear. The unlearning algorithm for the aforementioned learning algorithm is described in Algorithm 4.3. The algorithm's inputs are the coded dataset $\{\overline{\boldsymbol{X}}, \overline{\boldsymbol{y}}\}$, the samples to be unlearned $[\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}]$, their indices $\boldsymbol{i}$ in the uncoded training dataset $[\boldsymbol{X}, \boldsymbol{y}]$, the matrix $\boldsymbol{G}$, and the original model estimated coefficients $\boldsymbol{W}^*$. The algorithm's outputs are the updated coded dataset $\{\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{y}}\}$, and the updated estimated coefficients of the model $\widetilde{\boldsymbol{W}}^*$. Essentially, the algorithm needs to identify the uncoded shards that include the samples with indices $\boldsymbol{i}$ as well as their corresponding coded shards using the matrix $\boldsymbol{G}$. The samples first need to be removed from the coded shard by subtracting them from the corresponding coded samples in all coded shards where they appear to eliminate their effect from the coded shards. Once all the coded shards are updated, they are then used to update their corresponding weak learners to unlearn these samples from the weak learner models followed by updating the final aggregate model using the updated weak learners' estimates. The following lemma proves that the algorithm guarantees perfect unlearning.

Lemma 4.3.1 *The unlearning algorithm described in Algorithm 4.3 perfectly unlearns the desired samples from the model in the sense of Definition 4.1.*

Proof: Without loss of generality, we consider a single sample $[\boldsymbol{x}_i^T, y_i]$ that is requested to be unlearned from the model, which appears in $[\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{y}}_j]$ that is used to train the j-th weak learner whose model is denoted by $\mathcal{M}_j^{\boldsymbol{\theta}}(\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{y}}_j)$. First, the algorithm updates this training dataset to be $[\widetilde{\boldsymbol{X}}_j, \widetilde{\boldsymbol{y}}_j]$ by subtracting the sample $[\boldsymbol{x}_i^T, y_i]$ from the corresponding coded sample in order to remove it from the dataset $[\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{y}}_j]$. Then, the j-th weak learner, whose new training dataset is $[\widetilde{\boldsymbol{X}}_j, \widetilde{\boldsymbol{y}}_j]$, is trained from scratch and the resulting model is denoted as $\mathcal{U}(\mathcal{M}_j^{\boldsymbol{\theta}}(\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{y}}_j), [\boldsymbol{x}_i^T, y_i])$. This model is equivalent to a model $\mathcal{M}_j^{\boldsymbol{\theta}'}(\widetilde{\boldsymbol{X}}_j, \widetilde{\boldsymbol{y}}_j)$, where $\boldsymbol{\theta}'$ is chosen randomly. Using the uniqueness property of the linear and ridge regression solutions, we have the following:

$$\mathcal{U}(\mathcal{M}_j^{\boldsymbol{\theta}}(\overline{\mathbf{X}}_j, \overline{\boldsymbol{y}}_j), [\boldsymbol{x}_i^T, y_i]) = \mathcal{M}_j^{\boldsymbol{\theta}''}(\overline{\mathbf{X}}_j \setminus \boldsymbol{x}_i, \overline{\boldsymbol{y}}_j \setminus y_i), \tag{4.3.5}$$

for some random $\boldsymbol{\theta}''$. Hence, the desired sample is perfectly unlearned from the j-th weak learner. The same argument applies to all other affected weak learners after removing the desired samples from their corresponding training datasets. Therefore, as the resulting models from the affected weak learners are updated along with re-calculating the aggregation function, the overall updated model perfectly unlearns the desired samples from the model. ■

For large-scale problems, we can speed up the unlearning algorithm even more. Using iterative optimization methods, one can start the optimization problem for the weak learners on the new training dataset using the solution from their previous model. Specifically, for linear and ridge regression problems the resulting model will always be the same as the one trained from scratch since these iterative methods will converge to a unique global minimizer regardless of the initialization. However, this cannot be used for other complex models such as the over-parameterized neural networks (NN) since the training loss can be zero for these models. Hence, when a sample is removed and the model is initialized from the previous model, it will immediately converge since the training loss is already zero, but this solution was reached in part due to the removed sample. Therefore, this approach in the over-parameterized scenario does not perfectly unlearn the sample.

The last design parameter of the coded learning is the generator matrix $\boldsymbol{G}$. One of the properties of the matrix $\boldsymbol{G}$ used in the encoder is its density ρ_G, and it can be seen in Algorithm 4.3 that the density of $\boldsymbol{G}$ directly affects the unlearning cost. For example, a sample whose corresponding row in $\boldsymbol{G}$ is dense requires updating more weak learners than a sample whose corresponding row is sparse. Therefore, the design of such a matrix is directly related to the efficiency of unlearning. If we aim to have the lowest unlearning cost for an encoder with a specific rate, we use the minimum matrix density that satisfies both of the aforementioned conditions for the matrix $\boldsymbol{G}$, which is $\rho_G = \frac{1}{n_r}$. This corresponds to the case where there is only one nonzero element in each row of the matrix $\boldsymbol{G}$, i.e., each sample only shows up exactly in one coded shard.

Remark. Since the choice of the encoder in Algorithm 4.1 is independent of the data, it does not leak any information about the data itself and does not affect the perfect unlearning condition. However, other types of data-dependent encoders may require additional steps to ensure the removal of the unlearned samples from the encoder itself, which may introduce additional overhead. An example of such encoders is one that assigns samples to weak learners based on some properties of the training dataset itself. This leakage of information, even if small, needs to be taken into account when designing perfect unlearning algorithms.

4.4 Experimental Results

In this section, we present simulation results of some experiments to compare the performance versus the unlearning cost on realistic and synthetic datasets for two algorithms: the uncoded machine unlearning algorithm described in [2] and the proposed coded machine unlearning algorithm. The experiments simulate the unlearning of a sample from the training dataset, where the performance is measured in terms of the mean squared error and the unlearning cost is measured in terms of the time required to retrain the affected weak learner.

We utilize the `sklearn.linear_model` package [75], specifically, `LinearRegression` or `Ridge` modules, to produce the simulation results for all the experiments. Since the cost of unlearning is related to the size of the shards, we sweep the variable n_s while fixing the rate for the coded scenario and observe the performance. Each point in the plots shows the average of a number of runs, where each run simulates the experiment on a randomly shuffled dataset that is then split into training and testing datasets according to the specified sizes. During each run, after splitting the dataset into training and testing, Algorithm 4.1 is run first using n_s and n_r for a specific code with rate $R_c = \frac{n_s}{n_r}$ and density $\rho_G = \frac{1}{n_r}$. Once the model is trained, a random sample from the training dataset is chosen to be unlearned using Algorithm 4.3. After all the runs are done, the performance is measured as the average mean squared error of the testing dataset, while the unlearning cost is measured as the average time required to retrain the affected weak learners, since removing a sample from the dataset has negligible cost.

For the simulations, datasets are preprocessed as follows, each column of the original feature matrix and the response vector is normalized to be in the range $[0, 1]$. If the random projections approximation [71] is used as described in (4.2.7), then the projections are done on the normalized features using a cosine activation function and the following parameters

$$\mathbf{\Phi}_i \sim \mathcal{N}(0, \frac{1}{2d} I_d), \tag{4.4.1}$$

$$\varsigma_i \sim \text{unif}(-\pi, \pi). \tag{4.4.2}$$

4.4.1 Results

We conduct three experiments to evaluate the proposed algorithm on realistic datasets. The first dataset is known as the Physicochemical Properties of Protein Tertiary Structure dataset [3]. The goal is to use the 9 original features to estimate the root mean square deviation. This dataset

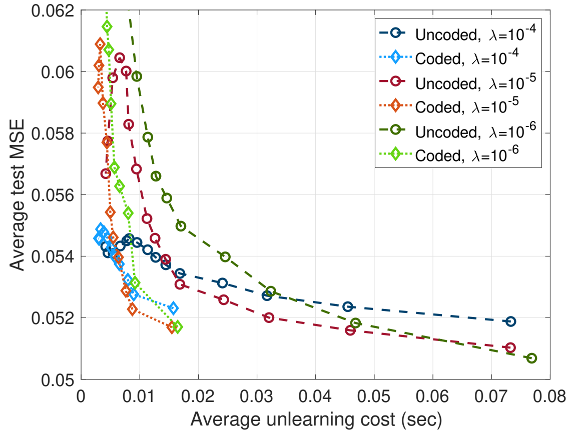


Figure 4.3: Performance vs unlearning cost for different values of λ using the Physico-chemical Properties of Protein Tertiary Structure dataset [3], random projections of features to a 300-dimensional space, and a code of rate $R_c = 5$.

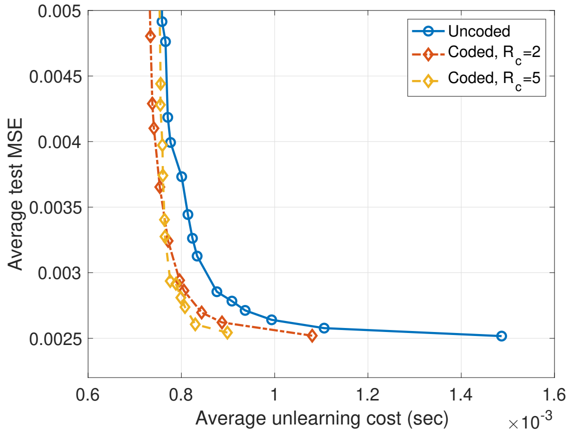


Figure 4.4: Performance vs unlearning cost for different rates R_c using the Computer Activity dataset [4], random projections of features to a 25-dimensional space, and $\lambda = 10^{-3}$.

includes $45{,}730$ samples, where $42{,}000$ samples are for training, and the rest are for testing. We consider random projections with $D = 300$. The results shown in Figure 4.3 show the simulation results for multiple values of $\lambda = 10^{-4}, 10^{-5}, 10^{-6}$ using a code of rate $R_c = 5$. It can be seen that coding provides better performance compared to the uncoded machine unlearning at lower unlearning cost, even when using regularization with different values.

The second dataset is known as the Computer Activity dataset [4]. It is concerned with estimating the portion of time that the CPU operates in user mode using different observed performance measures. We consider random projections of the original features to a space with $D = 25$. The dataset has $8{,}192$ samples, with 12 original features, of which $7{,}500$ samples are for training while the rest are for testing. The experiments use a regularization parameter $\lambda = 10^{-3}$ and different code rates $R_c = 2, 5$ and the results are shown in Figure 4.4. In this figure, we observe that coding provides better performance compared to the uncoded machine unlearning at a lower unlearning cost. Additionally, different rates allow for different achievable performance measures as evident in the figure. More on the effect of code rates will be discussed later in the experiments on a large-scale synthetic dataset.

Finally, we experiment on the Combined Cycle Power Plant dataset [3]. The goal is to estimate the net hourly electrical energy output using different ambient variables around the plant. The dataset has $9{,}568$ samples, with 4 original features, of which $9{,}000$ samples are for training while the rest are for testing. We consider random projections with $D = 20$. The experiments use linear regression with no regularization and different code rates $R_c = 2, 5$, and their results are shown in Figure 4.5. For this case, there is no region of coding to operate in, and intuitively, we do not expect it to beat the performance of the original learning algorithm with a single uncoded shard. However, although coding does not provide a better trade-off in this case, it does not exhibit a worse trade-off either.

The above experiments show results for datasets with relatively small to moderate size and number of features. It remains to be seen if similar behavior can be observed if the dataset size, as well as the number of features, become much larger. The following experiment shows simulation results of a synthetic dataset generated as follows: a total of $600{,}000$ samples are generated, each with i.i.d. features of size $d = 100$ drawn from lognormal distribution with parameters $\mu = 1, \sigma^2 = 4$, then passed through a random 3-hidden-layers NN, followed by an output layer with standard normal noise term to generate the desired response variable. The layers contain $50, 25, 50$ nodes, respectively, with a sigmoid activation function and their weights and biases are i.i.d. drawn from the standard normal distribution. We use $\lambda = 10^{-2}$ and apply random projections on the original features using the parameters described above and $D = 2{,}000$. The dataset is split into $500{,}000$ samples for training and $100{,}000$ for testing. The simulation results are shown in Figure 4.6. Note that log-scale is used on the x-axis for better showing of the curves for the coded scenarios. It

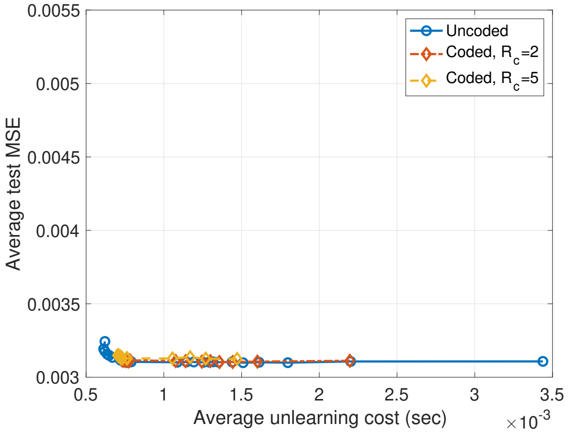


Figure 4.5: Performance vs unlearning cost for different rates R_c using the Combined Cycle Power Plant dataset [3] and random projections of features to a 20-dimensional space.

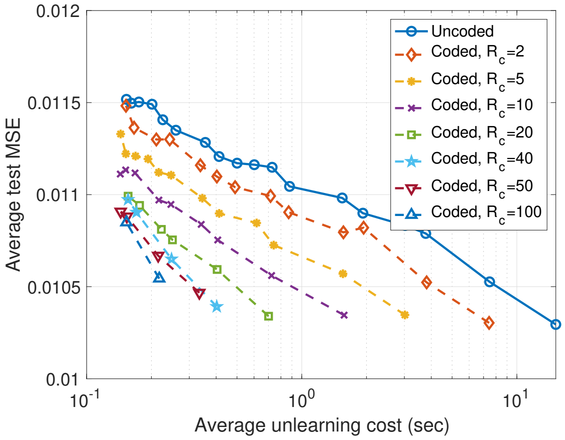


Figure 4.6: Performance vs unlearning cost for synthetic data generated from an NN with $\mathrm{lognormal}(1, 4)$ features using random projections to a 2,000-dimensional space, $\lambda = 10^{-2}$, and codes of different rates R_c.

can be observed that as we increase the rate of the code, the unlearning cost decreases while the minimum achievable MSE increases. Hence, in practical settings, one can choose the maximum code rate that satisfies a performance comparable to the original learning algorithm.

4.4.2 Discussion

The success of the proposed algorithm is prominently seen in cases where the baseline uncoded machine unlearning algorithm exhibits significant degradation in performance as unlearning cost decreases. One possible intuition into why this phenomenon occurs is related to the samples used for training each of the weak learners. Influential samples have been explored in the literature extensively [76]. As we previously discussed, coding is a method of combining samples into the coded dataset, including these influential samples.

Let us examine the influence of individual samples on the performance of the trained model. We take two of the previously considered datasets, the Computer Activity dataset [4] and the

Combined Cycle Power Plant dataset [3]. We conduct the following experiment: we randomly shuffle the data and split it into training and testing datasets with the same sizes as we used before, then we remove samples from the training dataset according to some criteria, then train a single learner on the remaining samples and observe its test MSE. We use two criteria of removal, the first is as follows: remove a sample if any of its original features lie outside certain percentiles. This criterion removes what we denote as outliers. The other criterion is as follows: remove samples whose original features lie inside certain percentiles, this removes what we denote as inliers. In other words, outliers are samples at the tails of the probability distribution function (PDF), and inliers are the ones close to the median. We vary these percentiles symmetrically on both ends of the PDF and observe the performance on the testing data for multiple runs then compute the observed average test MSE. Figure 4.7 shows the experiment results for the dataset with Computer Activity dataset and Figure 4.8 shows the experiment results for the Combined Cycle Power Plant dataset. In Figure 4.7, we see a degradation in performance as more outlier samples are removed which is much more significant and immediate compared to the case where inlier samples are removed. On the other hand, in Figure 4.8, the performance of the model after removing outliers and inliers is quite similar until we remove more than 50% of the samples, then a small gap appears between the two curves that gets larger as the number of removed samples increases.

We believe that one explanation behind the behavior of the uncoded machine unlearning is related to the existence of these influential samples, i.e., outlier samples. In particular, if influential samples exist in the dataset, then the uncoded machine unlearning suffers significant degradation as we increase the number of shards and the proposed algorithm can provide a better performance vs cost trade-off. On the other hand, if such influential samples do not exist, then the uncoded machine unlearning does not exhibit any degradation in performance as the number of shards in-

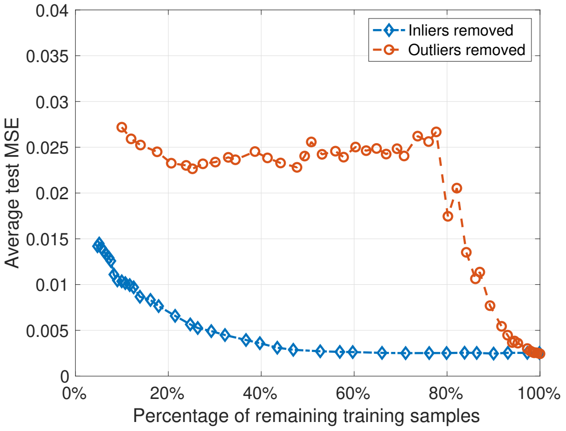


Figure 4.7: Original learning algorithm's performance vs percentage of remaining samples after removal of outliers and inliers from the Computer Activity dataset [4].

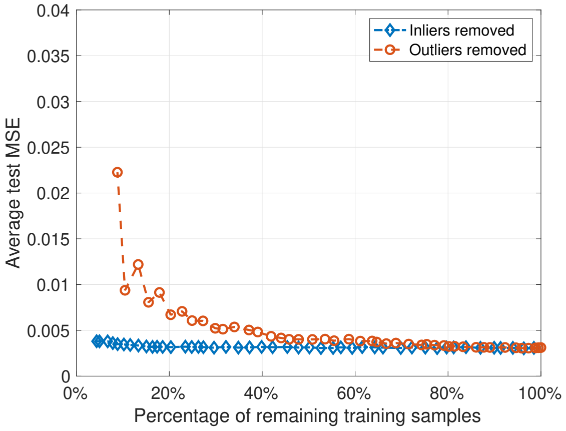


Figure 4.8: Original learning algorithm's performance vs percentage of remaining samples after removal of outliers and inliers from the Combined Cycle Power Plant dataset [3].

creases, and as shown in the experiment in Figure 4.5, the proposed algorithm does not improve on the uncoded machine unlearning and does not negatively affect it either. It is worth noting that these influential samples exist in heavy-tailed distributions which are quite common in a range of real-world examples such as technology, social sciences and demographics, medicine, etc, where machine learning is increasingly employed for these applications. In the aforementioned experiments, we observed that if the probability distribution functions of some of the features have heavy tails, then we have a trade-off for the uncoded machine unlearning and coding provides gain in such case. However, if there are no heavy tails in the probability distribution functions then we do not see this trade-off and, hence, coding does not provide improvements nor deterioration.

To verify this observation, we create three synthetic datasets with known feature distribution and known relationships to the response variable. Each one of the datasets has $d = 100$ i.i.d. feature vectors whose elements are drawn from $\mathrm{lognormal}(\mu, \sigma^2)$ distribution to create the feature matrix $\boldsymbol{X}$. Then, we map these features $\boldsymbol{X}$ to a degree 3 polynomial with no interaction terms, resulting in the following

$$\boldsymbol{X}_p = [\boldsymbol{X}, \boldsymbol{X}^{\circ 2}, \boldsymbol{X}^{\circ 3}]. \tag{4.4.3}$$

The response variable is generated using (4.2.8) with i.i.d. elements of $\boldsymbol{w}$ and $\boldsymbol{n}$ drawn from the standard normal distribution. The lognormal distribution has two parameters, μ and σ^2. We fix $\mu = 1$ and vary σ^2. As σ^2 increases, the tail becomes heavier; hence, we expect the trade-off to be more evident. The number of samples in each dataset is $25{,}000$ samples, of which $23{,}000$ are used for training and the rest are used for testing. The simulated experiments for $\sigma^2 = 0.1, 0.5, 0.7$ are shown in Figure 4.9, where the code used for all datasets has rate $R_c = 5$. As can be seen from the figure, as we increase the value of σ^2, the tail becomes heavier and the trade-off becomes more significant for the uncoded machine unlearning algorithm. Additionally, as the tail becomes heavier, the gain provided by the proposed algorithm in terms of the trade-off is more significant compared to the uncoded machine unlearning.

Additionally, we run experiments showing the effect of removing inliers versus removing outliers, for these synthetic datasets. We run the inlier and outlier removal process based on the original features and observe the performance of the trained model's performance on the testing dataset. Figure 4.10 shows results of these experiments. Similar to what we observed in the realistic dataset experiments, for distributions with heavier tails, removing outlier samples has more influence than inlier samples in terms of performance.

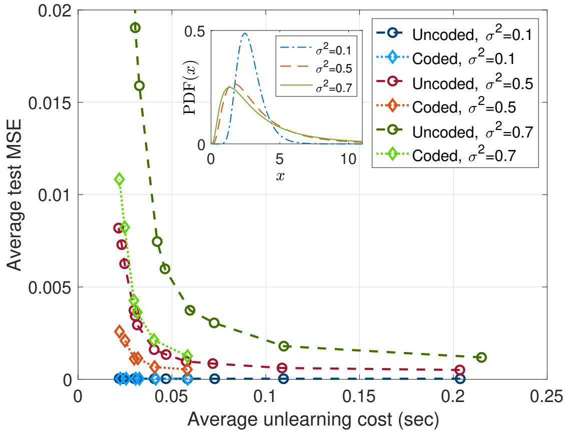


Figure 4.9: Performance vs unlearning cost for synthetic data with lognormal features with fixed $\mu = 1$ and different values of σ^2 used in a polynomial of degree 3. The rate of the code is $R_c = 5$. The inset figure shows the PDFs of the original lognormal features of the considered datasets with $\mu = 1$ and different values of σ^2.

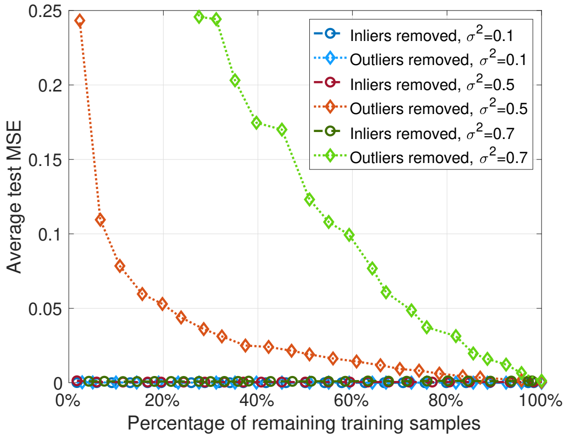


Figure 4.10: Original learning algorithm's performance vs percentage of remaining samples after removal of outliers and inliers from the lognormal polynomial datasets.

4.5 Conclusion

This chapter presented an efficient machine unlearning algorithm based on random coding. It focuses on the trade-off between performance and unlearning cost to compare unlearning algorithms. The proposed algorithm applies an encoding process of samples prior to training an ensemble learning model for regression problems. A handful of experiments were conducted to examine the behavior of the proposed algorithm against the uncoded baseline algorithm on multiple synthetic and realistic datasets. The proposed algorithm succeeds in providing a better trade-off for various realistic datasets with different values of the underlying parameters. On the other hand, datasets for which the uncoded machine unlearning algorithm does not exhibit any trade-off between performance and unlearning cost were considered, and it was shown experimentally that coding in these scenarios maintains performance on par with the uncoded machine unlearning algorithm. Experiments showed that when using appropriate codes, one can potentially reduce the unlearning cost to a fraction of the unlearning cost for a single learner trained on the entire dataset while observing a comparable performance. Finally, some discussions are provided on whether we should expect the proposed algorithm to outperform the uncoded machine unlearning algorithm based on the existence of influential samples in the dataset using properties of the probability distribution function of the dataset features.

CHAPTER 5

Federated Learning with Opt-Out Differential Privacy

5.1 Introduction

The abundance of data and advances in computation infrastructure have enabled the training of high-quality machine learning models. On the other hand, the data is distributed over many devices that are typically power-constrained and have limited computational capabilities. To reduce the amount of data transmission over networks and maintain the privacy of raw data, [33] proposed the federated learning framework for training a central server-side model using decentralized data at clients. Federated learning frameworks aim to train a global model iteratively and collaboratively using clients' data. During each round, the server has access to a select number of clients, each of whom has a local dataset. The server broadcasts the current model to such clients, who train the model by taking gradient steps using their local data on the model and returning the gradient-based update back to the server. The server then aggregates the updates and produces the new global model for the next round. Despite the clients' data being kept on device in federated learning, the deployed model at the central server is still vulnerable to various privacy attacks, such as membership inference attacks, model inversion attacks, and others. Utilizing differential privacy in federated learning provides a suitable solution to protect the models from privacy attacks.

This chapter considers privacy heterogeneity in federated learning setups. We propose a new federated learning setup where privacy is enabled by default for all clients, while giving clients the option to opt out of privacy. Moreover, we present a federated learning algorithm for this setup, called *FeO2*, and study its performance compared to the baseline private algorithm. The optimality of the proposed algorithm in the simplified setup of federated linear regression is shown, where an incentive for opting out of privacy is also shown. Experiments on real-world and synthetic federated datasets are conducted to examine the performance of the proposed algorithm and compare it to the baseline private algorithm where gains in the global model performance, as well as the local models' performances, are observed.

5.2 Federated Learning and Privacy

During each communication round t, the server sends the current model state, i.e., $\boldsymbol{\theta}^t$, to the set of available clients during that round, denoted by $\mathcal{C}^t$, who train the model on their local datasets to minimize their local loss functions $\ell_i(\cdot)$. The clients then return the updated model to the server who aggregates them, e.g., averages them, to produce the next model state $\boldsymbol{\theta}^{t+1}$. One approach to privacy-preserving FL algorithms utilizes differential privacy to provide privacy guarantees. Differential privacy is a widely studied and accepted mathematical notion that describes privacy-preserving algorithms where the information leakage of private data is bounded. Differential privacy is defined as follows

Definition 5.1 (differential privacy (DP) [77]) *A randomized algorithm $A(\cdot)$, whose image is de-*

noted as $\mathbf{O}$*, is said to be* (ϵ, δ)*-DP if for any two inputs* $\mathbf{D}$ *and* $\mathbf{D}'$ *that differ in just one entry, and all subsets* $O \subseteq \mathbf{O}$ *the following relationship holds*

$$\Pr(A(\boldsymbol{D}) \in O) \leqslant e^{\epsilon} \Pr(A(\boldsymbol{D}') \in O) + \delta. \tag{5.2.1}$$

In federated learning, instead of targeting the privacy of individual samples of a client, one can apply client-level differential privacy by having the adjacent datasets describe the case where the data removed is all the samples associated with a single client [39]. It is worth noting that using differential privacy in federated learning causes unavoidable degradation in performance relative to the guaranteed privacy parameters.

We examine heterogeneity in privacy requirements in federated learning setups. We present a new setup of privacy-preserving federated learning where privacy is enabled by default and the clients may choose to opt out. We desire to understand the implications of the setting where a fraction of the clients choose to opt out of privacy, even if they represent a small percentage of the overall population, to improve the performance of the global model as well as the personalized local models.

Next, we describe an approach for privacy-preserving federated learning as well as an approach to personalized federated learning. The proposed algorithm *FeO2* employs a version of both approaches, and they are considered the two baselines against which *FeO2* is examined.

5.2.1 Privacy-Preserving Federated Learning: *DP-FedAvg*

To design privacy-preserving federated learning algorithms using differential privacy, a few modifications are required to the baseline federated averaging algorithm [39]. Specifically, two modifications are introduced: clipping and noising. Considering client-level privacy, the averaging operation at the server is the target of such modifications. Assume clients are selected each round with probability q_s from the population of all clients of size N. First, each client update is clipped to have norm at most S, then the average is computed and additive Gaussian noise is added with mean zero and covariance $\sigma^2 I = z^2(\frac{S}{q_s N})^2 I$. The variable z is referred to as the noise multiplier, which dictates the achievable values of (ϵ, δ)-DP. Training the model via multiple rounds increases the amount of leaked information, the moment accountant method, introduced by [34], can be used to provide a tighter estimate of the resulting DP parameters (ϵ, δ) for the entire training process.

Selecting the clipping threshold, as well as the noise multiplier, is instrumental to getting useful models with desirable privacy guarantees. During training, the norm of updates can either increase or decrease, if the norm increases or decreases significantly compared to the clipping norm, the algorithm may slow down or diverge. Hence, [40] presented a solution to privately and adaptively update the clipping norm during each round of communication in federated learning based on the feedback from clients on whether or not their update norm exceeded the clipping norm. This method produces an effective noise multiplier from which one can compute the privacy loss. We consider this algorithm as the baseline for privacy-preserving federated learning and refer to it in the rest of the chapter as *DP-FedAvg*. The case where no noise is added is the baseline for the non-private federated learning algorithm, which is referred to simply as *non-private*.

5.2.2 Personalized Federated Learning: *Ditto*

There are multiple approaches to personalization in federated learning. In this chapter, we consider employing a recent work by [78], referred to as *Ditto*, due to its simplicity and modularity. *Ditto* is a bi-level optimization objective, which does not modify the global FL training process, but adds a local training task to personalize a model locally with the following local loss function:

$$\ell'_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}^*) = \ell_i(\boldsymbol{\theta}_i) + \frac{\lambda_i}{2}\|\boldsymbol{\theta}_i - \boldsymbol{\theta}^*\|^2, \tag{5.2.2}$$

where $\boldsymbol{\theta}_i$ is the personalized local model for client c_i, λ_i is the regularization parameter for client c_i, and $\boldsymbol{\theta}^*$ is the global server-side model. As *Ditto* can simply be added onto the previously described learning algorithms, we combine it with them to train personalized local models for each client. We utilize *Ditto* to train the personalized models for all algorithms in the experimental evaluation to examine the performance of such algorithms in terms of local models' performance. In particular, we demonstrate that *FeO2* with a *Ditto* add-on can learn good personalized local models compared to *DP-FedAvg*.

5.3 Federated Learning with Opt-Out Differential Privacy: *FeO2*

In this section, we describe the *FeO2* algorithm that is designed to take advantage of the aforementioned heterogeneous privacy setup. The proposed algorithm is described in Algorithm 5.1. As mentioned before, the *FeO2* algorithm utilizes differential privacy with adaptive clipping, and *Ditto*, optionally, for personalized learning.

First, we provide the notation of the variables in the algorithm. We split the set of clients $\mathcal{C}$ into two subsets containing private and non-private clients denoted by $\mathcal{C}_{\text{p}}$ and $\mathcal{C}_{\text{np}}$, respectively. Assume the number of private and non-private clients is $N_{\text{p}} = (1 - \rho_{\text{np}})N$ and $N_{\text{np}} = \rho_{\text{np}}N$, where ρ_{np} is

Algorithm 5.1 FeO2: Federated learning with opt-out DP

```
Inputs: model parameters $\boldsymbol{\theta}^0$, sensitivity $S^0$, learning rate $\eta$, personalized learning rate $\eta_p$, ratio $r$, noise multipliers $z, z_b$, quantile $\kappa$, and factor $\eta_b$.
Outputs: $\boldsymbol{\theta}^T, \{\boldsymbol{\theta_j}\}_{j \in [N]}$
At server:
for round $t = 0, 1, 2, ..., T-1$ do
    $\mathcal{C}^t \leftarrow$ Sample $N^t$ clients from $\mathcal{C}$
    for client $c_j$ in $\mathcal{C}^t$ in parallel do
        $\triangle\boldsymbol{\theta}_j^t, b_j^t \leftarrow$ ClientUpdate($\boldsymbol{\theta}^t, c_j, S^t$)
    end for
    $N_{\text{p}}^t \leftarrow |\mathcal{C}_{\text{p}}^t|, \quad N_{\text{np}}^t \leftarrow |\mathcal{C}_{\text{np}}^t|, \quad z^t \leftarrow z\frac{S^t}{N_{\text{p}}^t}$
    $\triangle\boldsymbol{\theta}_{\text{p}}^{t+1} \leftarrow \frac{1}{N_{\text{p}}^t}\sum_{i \in \mathcal{C}_{\text{p}}^t} \triangle\boldsymbol{\theta}_i^t + \mathcal{N}(\mathbf{0}, (z^t)^2\mathbf{I})$
    $\triangle\boldsymbol{\theta}_{\text{np}}^{t+1} \leftarrow \frac{1}{N_{\text{np}}^t}\sum_{i \in \mathcal{C}_{\text{np}}^t} \triangle\boldsymbol{\theta}_i^t$
    $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t + \frac{1}{N_{\text{np}}^t + rN_{\text{p}}^t}[N_{\text{np}}^t \triangle \boldsymbol{\theta}_{\text{np}}^{t+1}$
                                    $+rN_{\text{p}}^t \triangle \boldsymbol{\theta}_{\text{p}}^{t+1}]$
    $S^{t+1} \leftarrow S^t e^{-\eta_b\left((\frac{1}{N^t}\sum_{i \in \mathcal{C}^t} b_i^t + \mathcal{N}(0, z_b^2 \frac{1}{N^t}^2)) - \kappa\right)}$
end for

At client $c_j$:
ClientUpdate($\boldsymbol{\theta}^0, c_j, S$):
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^0$
$\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}^0$ (if not initialized)
$\mathcal{B} \leftarrow$ batch the client's data $\boldsymbol{D}_j$
for epoch $e = 1, 2, ..., E$ do
    for $B$ in $\mathcal{B}$ do
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla\ell_j(\boldsymbol{\theta}, B)$
        $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j - \eta_p(\nabla\ell_j(\boldsymbol{\theta}_j, B) + \lambda(\boldsymbol{\theta}_j - \boldsymbol{\theta}^0))$
        [Ditto: optional for personalization]
    end for
end for
$\triangle\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\theta}^0$
$b \leftarrow \mathbb{1}_{\|\triangle\boldsymbol{\theta}\|_2 \leqslant S}$
return Clip($\triangle\boldsymbol{\theta}, S$), $b$ to server

Clip($\boldsymbol{\theta}, S$):
    return $\boldsymbol{\theta} \times \frac{S}{\max(\|\boldsymbol{\theta}\|_2, S)}$ to client
```

the fraction of clients who opt out. The rest of the hyperparameters in the algorithm are as follows: the ratio hyperparameter r, the noise multipliers z, z_b, the clipping sensitivity S, the learning rate at clients η, the personalized learning rate at clients η_p, quantile κ, and factor η_b. The superscript $(\cdot)^t$ is used to denote a parameter during the t-th training round.

During round t of training, no additional steps are required for the clients during model training. Clients train the received model using their local data then send back their clipped updates $\Delta\boldsymbol{\theta}_j^t$ along with their clipping indicator b_j^t to the server. The server collects the updates from clients and performs a two-step aggregation process to the updates. During the first step, the updates from the non-private clients are passed through an averaging function to produce $\triangle\boldsymbol{\theta}_{\text{np}}^{t+1}$, while the updates from the private clients are passed through a differentially private-averaging function to produce $\triangle\boldsymbol{\theta}_{\text{p}}^{t+1}$. The second step of the aggregation is combining the outputs of the previous two averaging functions to produce the next iteration of the model. In this step, the server performs a weighted average of the two outputs. The weights for each are chosen based on the number of private and non-private clients, as well as a *ratio* hyperparameter r. The output of this step is $\triangle\boldsymbol{\theta}^{t+1}$, which is then added to the previous model state to produce the next model state.

The ratio hyperparameter r is chosen based on multiple factors, which we discuss next. In general, we desire the value of r be bounded as $0 \leqslant r \leqslant 1$ in *FeO2* to utilize non-private clients' updates more meaningfully. The first factor to consider when choosing r is related to the desired privacy budget, lower privacy budget ϵ requires more noise to be added, leading to a lower value of r. This intuition follows from observing Lemma 11 in [79] as will be shown in a simplified example in the next section. Another factor that is more difficult to quantify is the heterogeneity between the non-private set of clients and the private set of clients. We give an example to illustrate this idea, assume the model is being trained on the MNIST dataset where each client has samples of only one digit. Consider two scenarios: opt-out clients have uniform digits, and opt-out clients have the same digit. It can be argued that the ratio r, when every other hyperparameter is fixed, should be higher in the second scenario compared to the first; since contributions from private clients are more significant to the overall model in the second scenario than the first. An experiment will be conducted in the experiments section to show this observation.

As for the personalization part, we have the hyperparameter λ. *FeO2* differs from *Ditto* in a number of major ways. First, The server-side aggregation in *Ditto* is the vanilla *FedAvg*; however, in *FeO2* the server-side aggregation is not *FedAvg*, but rather a new aggregation rule which utilizes the privacy choices made by clients. Second, *Ditto* is designed for robustness against malicious clients; hence, the performance on malicious clients is not measured. That is not the case in *FeO2*, where measuring the performance for private and non-private clients is needed, and improving both is desired. Third, the server in *Ditto* is unaware of the status of the clients, i.e., whether or not they are malicious; while in *FeO2* the server is aware of the privacy choices made by clients which can be used during training to the advantage of the server.

5.4 Analyzing Federated Linear Regression

In this section, we provide some insights into the proposed *FeO2* algorithm through a simplified setup, known as the federated linear regression, inspired by the one proposed by [78]. We first start by considering the global estimation on the server and show that *FeO2* is Bayes optimal when using the appropriate value of the ratio r. Then we consider personalization using *Ditto* for both private and non-private clients and show that combining the optimal *FeO2* with *Ditto* is Bayes optimal for private and non-private clients when using appropriate values of the regularization parameters λ_{p} and λ_{np}. Finally, we show the gain of opting out in the proposed personalized setup compared to the baseline.

5.4.1 Setup

Assume the number of samples per client is fixed and the the effect of clipping is negligible. Let us denote the total number of clients as N, of whom $N_{\text{np}} = \rho_{\text{np}} N$ are non-private and $N_{\text{p}} = (1-\rho_{\text{np}})N$ are private. Denote the number of samples held by each client as m_s, the samples at client c_j as $[\boldsymbol{X}_j, \boldsymbol{y}_j]$, where $\boldsymbol{X}_j$ is the data features matrix, and $\boldsymbol{y}_j$ is the response vector. Let us denote the relationship between $\boldsymbol{y}_j$ and $\boldsymbol{X}_j$ as

$$\boldsymbol{y}_j = \boldsymbol{X}_j \boldsymbol{\phi}_j + \boldsymbol{n}_j \tag{5.4.1}$$

where the observations noise vector $\boldsymbol{n}_j \sim \mathcal{N}(0, \beta^2 I_{m_s})$, and $\boldsymbol{\phi}_j$ is the vector of length d to be estimated. The vector $\boldsymbol{\phi}_j$ is described as

$$\boldsymbol{\phi}_j = \boldsymbol{\phi} + \boldsymbol{p}_j \tag{5.4.2}$$

where $\boldsymbol{p}_j \sim \mathcal{N}(0, \tau^2 I_d)$, and $\boldsymbol{\phi}$ is the vector to be estimated at the server. τ^2 is a measure of relatedness, as specified by [78], to denote the non i.i.d. nature in the vectors. The loss function at client c_j is as follows

$$\ell_j(\boldsymbol{\phi}) = \frac{1}{m_s} \|\boldsymbol{X}_j \boldsymbol{\phi} - \boldsymbol{y}_j\|_2^2 \tag{5.4.3}$$

Local estimate of $\boldsymbol{\phi}_j$ at client c_j that minimizes the loss function given $\boldsymbol{X}_j$ and $\boldsymbol{y}_j$ is denoted by $\widehat{\boldsymbol{\phi}}_j$ and is computed as

$$\widehat{\boldsymbol{\phi}}_j = \big(\boldsymbol{X}_j^T \boldsymbol{X}_j\big)^{-1} \boldsymbol{X}_j^T \boldsymbol{y}_j, \tag{5.4.4}$$

which is distributed as $\widehat{\boldsymbol{\phi}}_j \sim \mathcal{N}\big(\boldsymbol{\phi}_j, \beta^2 (\boldsymbol{X}_j^T \boldsymbol{X}_j)^{-1}\big)$. Let us assume that $\boldsymbol{X}_j^T \boldsymbol{X}_j = n_s I_d$, then the loss function can be translated to

$$\ell_j(\boldsymbol{\phi}) = \frac{1}{2} \big\|\boldsymbol{\phi} - \frac{1}{n_s} \sum_{i=1}^{n_s} \boldsymbol{y}_{j,i} \big\|_2^2, \tag{5.4.5}$$

where $\boldsymbol{y}_{j,i}$'s are the noisy observations of the vector $\boldsymbol{\phi}_j$ at client c_j. The updates sent to the server by client c_j are as follows

$$\boldsymbol{\psi}_j = \widehat{\boldsymbol{\phi}}_j + \boldsymbol{l}_j \tag{5.4.6}$$

where $\boldsymbol{l}_j = 0$ for non-private clients or $\boldsymbol{l}_j \sim \mathcal{N}(\mathbf{0}, N_{\mathrm{p}}\gamma^2 I_d)$. Note that we still consider client-level privacy; however, we move the noise addition process from the server side to the client side such that when the server aggregates the private clients updates the resulting privacy noise covariance is equivalent to the desired value by the server, i.e., γ^2. This is done for the simplicity and clarity of the discussion.

In this setup, the problem becomes a vector estimation problem and the goal at the server is to estimate the vector $\boldsymbol{\phi}$ given the updates from all clients, denoted by $\{\boldsymbol{\psi}_i : i \in [N]\}$ as

$$\boldsymbol{\theta}^* := \arg\min_{\widehat{\boldsymbol{\theta}}} \left\{ \mathbb{E}\left[\frac{1}{2} \|\widehat{\boldsymbol{\theta}} - \boldsymbol{\phi}\|_2^2 \middle| \boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_N \right] \right\}. \tag{Global Bayes objective}$$

On the other hand, client c_j's goal is to estimate the vector $\boldsymbol{\phi}_j$ given their local estimate $\widehat{\boldsymbol{\phi}}_j$ as well as the updates from all other clients $\{\boldsymbol{\psi}_i : i \in [N] \setminus j\}$ as

$$\boldsymbol{\theta}_j^* := \arg\min_{\widehat{\boldsymbol{\theta}}} \left\{ \mathbb{E}\left[\frac{1}{2} \|\widehat{\boldsymbol{\theta}} - \boldsymbol{\phi}_j\|_2^2 \middle| \{\boldsymbol{\psi}_i : i \in [N] \setminus j\}, \hat{\boldsymbol{\phi}}_j \right] \right\}. \tag{Local Bayes objective}$$

Now, considering the value of $\boldsymbol{\phi}$, the covariance matrix of client c_j's update is denoted by Σ_j

and is expressed as follows

$$\Sigma_j = \begin{cases} \beta^2(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1} + \tau^2 I_d, & \text{if } c_j \in \mathcal{C}_{\text{np}} \\ \beta^2(\boldsymbol{X}_j^T\boldsymbol{X}_j)^{-1} + (\tau^2 + N_{\text{p}}\gamma^2) I_d, & \text{if } c_j \in \mathcal{C}_{\text{p}} \end{cases} \tag{5.4.7}$$

We have $\boldsymbol{X}_j^T\boldsymbol{X}_j = n_s I_d$, and let $\frac{\beta^2}{n_s} = \alpha^2$, then we have

$$\Sigma_j = \begin{cases} (\alpha^2 + \tau^2) I_d, & \text{if } c_j \in \mathcal{C}_{\text{np}} \\ (\alpha^2 + \tau^2 + N_{\text{p}}\gamma^2) I_d, & \text{if } c_j \in \mathcal{C}_{\text{p}} \end{cases} \tag{5.4.8}$$

$$= \begin{cases} \sigma_c^2 I_d, & \text{if } c_j \in \mathcal{C}_{\text{np}} \\ \sigma_p^2 I_d, & \text{if } c_j \in \mathcal{C}_{\text{p}} \end{cases} \tag{5.4.9}$$

Next, we discuss the optimality of *FeO2* for the specified federated linear regression problem for the server's global model as well as the clients' personalized local models using a variation of *Ditto* in our setup.

5.4.2 Global Model On The Server

In the considered federated setup, the server aims to find $\widehat{\boldsymbol{\theta}}^*$ described as follows

$$\widehat{\boldsymbol{\theta}}^* := \arg\min_{\widehat{\boldsymbol{\theta}}} \left\{ \frac{1}{2} \left\| \sum_{i\in[N]} w_i \psi_i - \widehat{\boldsymbol{\theta}} \right\|_2^2 \right\}. \tag{Global FeO2 objective}$$

The server's goal is to combine the client updates such that the estimation error of $\boldsymbol{\phi}$, described in (Global Bayes objective), is minimized. For the considered setup, the server aims to utilize the updates sent by clients, i.e., $\{\boldsymbol{\psi}_i : i \in [N]\}$, to estimate the vector $\boldsymbol{\phi}$. The estimate at the server is denoted by $\boldsymbol{\theta}$. First, we state an important lemma that will be used throughout this section.

Lemma 5.4.1 (Lemma 2 in [78]) *Let $\boldsymbol{\phi}$ be drawn from the non-informative uniform prior on $\mathbb{R}^d$. Also, let $\{\boldsymbol{\psi}_i : i \in [N]\}$ denote noisy observations of $\boldsymbol{\phi}$ with independent additive zero-mean independent Gaussian noise and corresponding covariance matrices $\{\Sigma_i : i \in [N]\}$. Let*

$$\Sigma_{\boldsymbol{\phi}} = \Big(\sum_{i\in[N]} \Sigma_i^{-1}\Big)^{-1}. \tag{5.4.10}$$

Then, conditioned on $\{\boldsymbol{\psi}_i : i \in [N]\}$, we have

$$\boldsymbol{\phi} = \Sigma_{\boldsymbol{\phi}} \sum_{i\in[N]} \Sigma_i^{-1}\boldsymbol{\psi}_i + \boldsymbol{p}_{\boldsymbol{\phi}}, \tag{5.4.11}$$

where $\boldsymbol{p}_{\boldsymbol{\phi}} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_{\boldsymbol{\phi}})$, which is independent of $\{\boldsymbol{\psi}_i : i \in [N]\}$.

Next, we state the Bayes optimality of the solution to the global FeO2 objective.

Lemma 5.4.2 (Global estimate optimality) *The solution to the global FeO2 objective from the server's point of view, with weights w_j's chosen below, is Bayes optimal in the considered federated linear regression problem.*

$$w_j = \begin{cases} \frac{1}{N_{np}+N_p r^*}, & \text{if } c_j \in \mathcal{C}_{np} \\ \frac{r^*}{N_{np}+N_p r^*}, & \text{if } c_j \in \mathcal{C}_p \end{cases} \tag{5.4.12}$$

where

$$r^* = \frac{\sigma_c^2}{\sigma_c^2 + N_p\gamma^2}. \tag{5.4.13}$$

Furthermore, the covariance of the estimation error is:

$$\Sigma_{s,\text{opt}} = \frac{1}{N}\left[\frac{\sigma_c^2(\sigma_c^2 + N_p\gamma^2)}{\sigma_c^2 + \rho_{np}N_p\gamma^2}\right] I_d. \tag{5.4.14}$$

Proof: First, for the considered setup, Lemma 5.4.1 states that the optimal aggregator at the server is the weighted average of the client updates. The server observes the updates $\{\boldsymbol{\psi}_i : i \in [N]\}$, which are noisy observations of $\boldsymbol{\phi}$ with zero-mean Gaussian noise with corresponding covariance matrices $\{\Sigma_i : i \in [N]\}$. Then, the server computes its estimate $\boldsymbol{\theta}$ of $\boldsymbol{\phi}$ as

$$\boldsymbol{\theta} = \Sigma_{\boldsymbol{\theta}} \sum_{i\in[N]} \Sigma_i^{-1}\boldsymbol{\psi}_i + \boldsymbol{p}_{\boldsymbol{\theta}}, \tag{5.4.15}$$

$$\tag{5.4.16}$$

where $\boldsymbol{p}_{\boldsymbol{\theta}} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{\theta}})$ and

$$\begin{aligned}\Sigma_{\boldsymbol{\theta}} = \Big(\sum_{i\in[N]} \Sigma_i^{-1}\Big)^{-1} &= \Big(N_{\text{np}}(\sigma_c^2 I_d)^{-1} + N_{\text{p}}(\sigma_p^2 I_d)^{-1}\Big)^{-1} &(5.4.17)\\ &= \frac{1}{N}\left[\frac{\sigma_c^2(\sigma_c^2 + N_{\text{p}}\gamma^2)}{\sigma_c^2 + \rho_{\text{np}}N_{\text{p}}\gamma^2}\right] I_d. &(5.4.18)\end{aligned}$$

In *FeO2*, we only have a single hyperparameter to tune, which is the ratio r. To achieve the same noise covariance as in (5.4.18) we need to choose the ratio r carefully. To this end, setting $r = \frac{\sigma_c^2}{\sigma_c^2 + N_{\text{p}}\gamma^2}$ in *FeO2* results in additive noise in the estimate with zero mean and covariance matrix

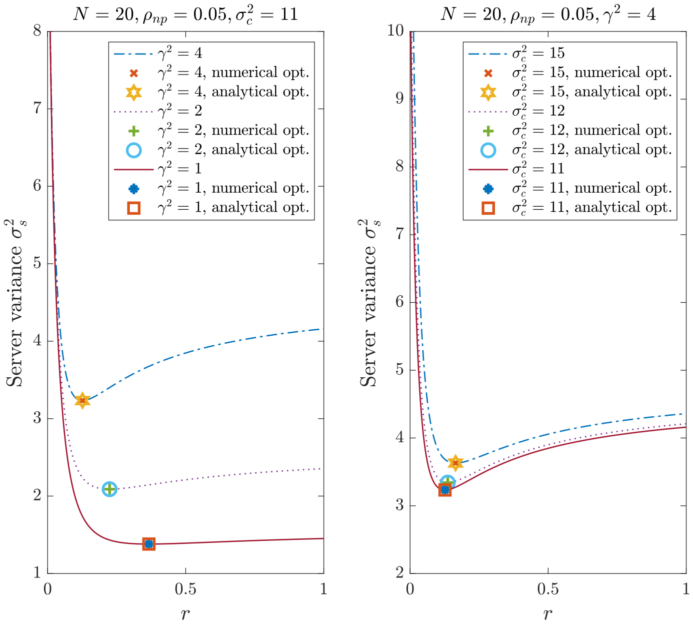


Figure 5.1: Server noise variance σ_s^2 vs the ratio hyperparameter r assuming $d = 1$. (left) trade-off for three values of γ^2, (right) trade-off for three values of σ_c^2.

as follows

$$\Sigma_{s,opt} = \frac{1}{N}\left[\frac{\sigma_c^2(\sigma_c^2 + N_{\text{p}}\gamma^2)}{\sigma_c^2 + \rho_{\text{np}}N_{\text{p}}\gamma^2}\right] I_d. \tag{5.4.19}$$

Therefore, the weighted average of the updates using the above weights results in the solution being Bayes optimal, i.e., produces $\boldsymbol{\theta}^*$. ■

Next, we simulate the server noise variance σ_s^2 against the ratio r for $d = 1$ and different values of σ_c^2 and γ^2 with N clients and ρ_{np} opt-put fraction. The results are shown in Figure 5.1, and we can see that the optimal ratio r^* in Lemma 5.4.2 minimizes the server estimation variance as expected.

Moreover, the server noise σ_s^2 vs the fraction of non-private clients ρ_{np} is compared for two scenarios using $d = 1$. The first is the baseline *FedAvg*, and the second is the optimal *FeO2*. We can see in Figure 5.2 that *FeO2* provides better noise variance at the server compared to *FedAvg*,

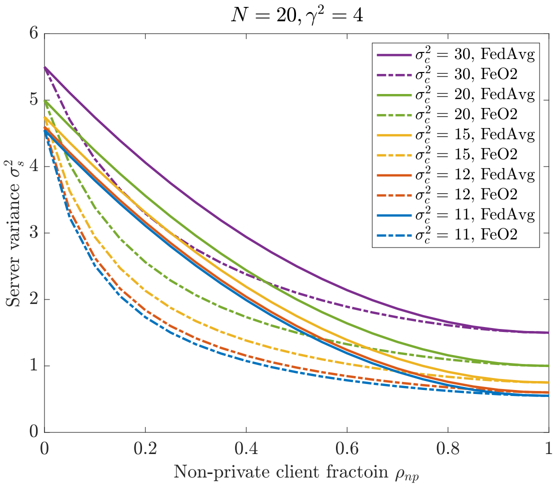


Figure 5.2: Server noise variance σ_s^2 vs non-private client fraction ρ_{np} assuming $d = 1$ for the baseline *FedAvg* aggregator and optimal *FeO2* aggregator.

and the gain can be significant for some values of ρ_{np}.

It is worth noting that the resulting server noise covariance when *FeO2* algorithm is used with vanilla *FedAvg*, i.e., $r = 1$ in the Algorithm 5.1 is

$$\Sigma_{s,\text{fedavg}} = \frac{1}{N}(\sigma_c^2 + (1 - \rho_{\text{np}})N_p\gamma^2)I_d, \tag{5.4.20}$$

hence, we have the following lemma.

Lemma 5.4.3 (Performance gap between baselines and optimal *FeO2*) *The gap in server's mean squared error performance between* FeO2 *with* FedAvg *and the optimal* FeO2, *and the gap between* DP-FedAvg *and the optimal* FeO2 *for the federated linear regression problem, specified above, are as follows*

$$MSE_{s,fedavg} - MSE_{s,opt} = \frac{\rho_{np}(1 - \rho_{np})N_p^2\gamma^4}{N(\sigma_c^2 + \rho_{np}N_p\gamma^2)}d, \tag{5.4.21}$$

$$MSE_{s,dp\text{-}fedavg} - MSE_{s,opt} = \frac{N_p\gamma^2\rho_{np}(\sigma_c^2 + N_p\gamma^2)}{N(\sigma_c^2 + \rho_{np}N_p\gamma^2)}d. \tag{5.4.22}$$

Note that both (5.4.21) and (5.4.22) are positive. It can be seen that if the number of clients is large ($N \to \infty$), the gap approaches $(1-\rho_{\text{np}})^2\gamma^2$ and $(1-\rho_{\text{np}})\gamma^2$ in (5.4.21) and (5.4.22), respectively. This is expected since the noise in the observation itself decreases as the number of clients increase. On the other hand, if $\rho_{\text{np}} \to 0$ or $\rho_{\text{np}} \to 1$, the gap vanishes as expected. Furthermore, if the noise added for privacy γ^2 is large ($\gamma^2 \to \infty$), the gap become increasingly significant.

5.4.3 Local Models On Clients

As mentioned before, *FeO2* differs from *Ditto* in many ways. First, the global model aggregation is different, i.e., *FedAvg* was employed in *Ditto* compared to the 2-step aggregator in *FeO2*. Second, in *Ditto* measuring the performance only considers benign clients, while in *FeO2* it is important to measure the performance of both non-private and private clients, and enhancing both is desired. In this part, we focus on the personalization part for both sets of clients. The goal at clients is to find the Bayes optimal solution to the (Local Bayes objective). However, in the considered federated setup, clients don't have access to individual updates from other clients, but rather have the global estimate $\widehat{\boldsymbol{\theta}}^*$. So, the local *FeO2* objective using *Ditto* is

$$\widehat{\boldsymbol{\theta}}_j^* := \arg\min_{\widehat{\boldsymbol{\theta}}} \left\{ \frac{1}{2}\|\widehat{\boldsymbol{\theta}} - \widehat{\boldsymbol{\phi}}_j\|_2^2 + \frac{\lambda}{2}\|\widehat{\boldsymbol{\theta}} - \widehat{\boldsymbol{\theta}}^*\|_2^2 \right\}, \qquad \text{(Local FeO2 objective)}$$

where $\widehat{\boldsymbol{\theta}}^*$ is the solution to (Global FeO2 objective).

To assess the quality of this procedure, we first compute the Bayes optimal local estimate $\boldsymbol{\theta}_j^*$ of $\boldsymbol{\phi}_j$ for the local objective at client c_j. We consider client c_j, which can be either private or non-private, and compute their minimizer of (Local Bayes objective). In this case, the client is given all other clients' estimates $\{\boldsymbol{\psi}_i : i \in [N] \setminus j\}$ and has their own local estimate $\hat{\boldsymbol{\phi}}_j$. To this end, we utilize Lemma 5.4.1 to find the optimal estimate $\boldsymbol{\theta}_j^*$. Given the updates by all other clients $\{\boldsymbol{\psi}_i : i \in [N] \setminus j\}$, the client can compute the estimate $\boldsymbol{\phi}^{\setminus j}$ of the value of $\boldsymbol{\phi}$ as

$$\boldsymbol{\phi}^{\setminus j} = \Sigma_{\boldsymbol{\phi}^{\setminus j}} \Big(\sum_{i=[N]\setminus j} \Sigma_i^{-1} \boldsymbol{\psi}_i \Big) + \boldsymbol{p}_{\boldsymbol{\phi}^{\setminus j}}, \tag{5.4.23}$$

where $\boldsymbol{p}_{\boldsymbol{\phi}^{\setminus j}} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{\phi}^{\setminus j}})$ and

$$\Sigma_{\boldsymbol{\phi}^{\setminus j}} = \Big(\sum_{i=[N]\setminus j} \Sigma_i^{-1} \Big)^{-1}, \tag{5.4.24}$$

$$= \Big(m \frac{1}{\sigma_c^2} I_d + n \frac{1}{\sigma_p^2} I_d \Big)^{-1}, \tag{5.4.25}$$

$$= \frac{\sigma_c^2 \sigma_p^2}{n\sigma_c^2 + m\sigma_p^2} I_d, \tag{5.4.26}$$

where $n = N_{\text{p}} - 1, m = N_{\text{np}}$ if c_j is private, or $n = N_{\text{p}}, m = N_{\text{np}} - 1$ if c_j is non-private. Then, the client uses $\Sigma_{\boldsymbol{\phi}^{\setminus j}}$ and $\hat{\boldsymbol{\phi}}_j$ to estimate $\boldsymbol{\theta}_j^*$ as

$$\boldsymbol{\theta}_j^* = \Sigma_{\boldsymbol{\theta}_j^*} \Big((\Sigma_{\boldsymbol{\phi}^{\setminus j}} + \tau^2 I_d)^{-1} \boldsymbol{\phi}^{\setminus j} + (\sigma_c^2 - \tau^2)^{-1} \hat{\boldsymbol{\phi}}_j \Big) + \boldsymbol{p}_{\boldsymbol{\theta}_j^*}, \tag{5.4.27}$$

$$= \Sigma_{\boldsymbol{\theta}_j^*} \Bigg(\Big(\frac{n\sigma_c^2 + m\sigma_p^2}{\sigma_c^2\sigma_p^2 + \tau^2(n\sigma_c^2 + m\sigma_p^2)} \Big) \boldsymbol{\phi}^{\setminus j} + \frac{1}{\sigma_c^2 - \tau^2} \hat{\boldsymbol{\phi}}_j^* \Bigg) + \boldsymbol{p}_{\boldsymbol{\theta}_j^*}, \tag{5.4.28}$$

where $\boldsymbol{p}_{\boldsymbol{\theta}_j^*} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{\theta}_j^*})$ and

$$\Sigma_{\boldsymbol{\theta}_j^*} = \Bigg(\Big(\frac{\sigma_c^2\sigma_p^2 + \tau^2(n\sigma_c^2 + m\sigma_p^2)}{n\sigma_c^2 + m\sigma_p^2} I_d \Big)^{-1} + ((\sigma_c^2 - \tau^2) I_d)^{-1} \Bigg)^{-1}, \tag{5.4.29}$$

$$= \frac{(\sigma_c^2 - \tau^2)\big(\sigma_c^2\sigma_p^2 + \tau^2(n\sigma_c^2 + m\sigma_p^2)\big)}{\sigma_c^2\big(n\sigma_c^2 + (m+1)\sigma_p^2\big)} I_d. \tag{5.4.30}$$

We expand (5.4.28) as

$$\boldsymbol{\theta}_j^* = \frac{\sigma_c^2\sigma_p^2 + \tau^2(n\sigma_c^2 + m\sigma_p^2)}{\sigma_c^2(n\sigma_c^2 + (m+1)\sigma_p^2)}\hat{\boldsymbol{\phi}}_j + \frac{(\sigma_c^2 - \tau^2)\sigma_p^2}{\sigma_c^2(n\sigma_c^2 + (m+1)\sigma_p^2)}\sum_{\substack{i\in\mathcal{C}_{\text{np}}\\ i\neq j}}\boldsymbol{\psi}_i + \frac{(\sigma_c^2 - \tau^2)}{(n\sigma_c^2 + (m+1)\sigma_p^2)}\sum_{\substack{i\in\mathcal{C}_{\text{p}}\\ i\neq j}}\boldsymbol{\psi}_i + \boldsymbol{p}_{\boldsymbol{\theta}_j^*}. \tag{5.4.31}$$

This is the Bayes optimal solution to the local Bayes objective optimization problem for client c_j in (Local Bayes objective). Now, recall that in *FeO2*, the clients do not have access to individual client updates, but rather the global model. Therefore, the clients solve the *FeO2* local objective in (Local FeO2 objective). Given a value of λ_j and the global estimate $\hat{\boldsymbol{\theta}}^*$, the minimizer $\hat{\boldsymbol{\theta}}_j(\lambda_j)$ of (Local FeO2 objective) is

$$\hat{\boldsymbol{\theta}}_j(\lambda_j) = \frac{1}{1+\lambda_j}\left(\hat{\boldsymbol{\phi}}_j + \lambda_i\hat{\boldsymbol{\theta}}^*\right) \tag{5.4.32}$$

$$= \frac{1}{1+\lambda_j}\left(\frac{(N_{\text{np}} + N_{\text{p}}r) + \lambda_j i_j}{(N_{\text{np}} + N_{\text{p}}r)}\hat{\boldsymbol{\phi}}_j + \frac{\lambda_j}{(N_{\text{np}} + N_{\text{p}}r)}\sum_{\substack{i\in\mathcal{C}_{\text{np}}\\ i\neq j}}\boldsymbol{\psi}_i + \frac{\lambda_j r}{(N_{\text{np}} + N_{\text{p}}r)}\sum_{\substack{i\in\mathcal{C}_{\text{p}}\\ i\neq j}}\boldsymbol{\psi}_i\right), \tag{5.4.33}$$

where $i_j = 1$ if c_j is non-private or $i_j = r$ if c_j is private. Now, we are ready to state the Bayes optimality of the local FeO2 objective for optimal values λ_j^* for all clients.

Lemma 5.4.4 (Local estimates optimality) *The solution to the local FeO2 objective from the clients' point of view using λ_j^* chosen below, under the assumption of global estimate optimal-*

ity stated in Lemma 5.4.2, is Bayes optimal in the considered federated linear regression problem.

$$\lambda_j^* = \begin{cases} \frac{1}{\Upsilon^2}, & \textit{if } c_j \in \mathcal{C}_{np} \\ \frac{N+\Upsilon^2 N+\Gamma^2(N-N_p)}{\Upsilon^2(\Upsilon^2+1)N+\Upsilon^2\Gamma^2(N-N_p+1)+\Gamma^2}, & \textit{if } c_j \in \mathcal{C}_p \end{cases} \tag{5.4.34}$$

where $\Upsilon^2 = \frac{\tau^2}{\alpha^2}$ *and* $\Gamma^2 = \frac{N_p\gamma^2}{\alpha^2}$.

Proof: To prove this lemma, as shown in [78], we only need to find the optimal values of λ_j^* that minimize the following

$$\lambda_j^* = \arg\min_{\lambda} \mathbb{E}\big(\|\boldsymbol{\theta}_j^* - \hat{\boldsymbol{\theta}}_j(\lambda)\|_2^2 \big| \boldsymbol{\phi}^{\setminus j}, \hat{\boldsymbol{\phi}}_j\big) \tag{5.4.35}$$

for private and non-private clients. To compute the values of λ_j^*, we plug in the values of $\boldsymbol{\theta}_j^*$ from (5.4.31) and $\boldsymbol{\theta}_j(\lambda)$ in (5.4.33), which gives us the following

$$\lambda_1 = \frac{(N_{\text{np}} + N_{\text{p}}r)\big(n\sigma_c^4 + m\sigma_c^2\sigma_p^2 - \tau^2(n\sigma_c^2 + m\sigma_p^2)\big)}{(N_{\text{np}} + N_{\text{p}}r)\big(\sigma_c^2\sigma_p^2 + \tau^2(n\sigma_c^2 + m\sigma_p^2)\big) - i_j\sigma_c^2(n\sigma_c^2 + (m+1)\sigma_p^2)}, \tag{5.4.36}$$

$$\lambda_2 = \frac{(N_{\text{np}} + N_{\text{p}}r)(\sigma_c^2 - \tau^2)\sigma_p^2}{\sigma_c^2(n\sigma_c^2 + (m+1)\sigma_p^2) - (N_{\text{np}} + N_{\text{p}}r)(\sigma_c^2 - \tau^2)\sigma_p^2}, \tag{5.4.37}$$

$$\lambda_3 = \frac{(N_{\text{np}} + N_{\text{p}}r)(\sigma_c^2 - \tau^2)}{r(n\sigma_c^2 + (m+1)\sigma_p^2) - (N_{\text{np}} + N_{\text{p}}r)(\sigma_c^2 - \tau^2)}, \tag{5.4.38}$$

$$\text{and } \lambda_j^* = \frac{1}{3}(\lambda_1 + \lambda_2 + \lambda_3) \tag{5.4.39}$$

where $r = \frac{\sigma_c^2}{\sigma_p^2}$. For non-private client $c_j \in \mathcal{C}_{\text{np}}$, we have $i_j = 1, n = N_{\text{p}}$ and $m = N_{\text{np}} - 1$. Substituting in (5.4.39) gives the desired result in (5.4.34). For private client $c_j \in \mathcal{C}_{\text{p}}$, we have $i_j = r, n = N_{\text{p}} - 1$ and $m = N_{\text{np}}$. Setting $\Upsilon^2 = \frac{\tau^2}{\alpha^2}$ and $\Gamma^2 = \frac{N_{\text{p}}\gamma^2}{\alpha^2}$, and substituting in (5.4.39) gives the desired results in (5.4.34). As a result, the resulting $\hat{\boldsymbol{\theta}}_j(\lambda_j^*)$ is Bayes optimal. ■

Next, we provide a few examples of corner cases for both λ_{p}^* and λ_{np}^* for the considered linear regression setup:

- $r \to 1$, i.e., noise added for privacy is too small, $\lambda_{\mathrm{p}}^* \to \frac{1}{\Gamma^2}$ and $\lambda_{\mathrm{np}}^* = \frac{1}{\Gamma^2}$, as in *Ditto* with *FedAvg* and no malicious clients.
- $N_{\mathrm{p}} \to N$, i.e., all clients are private, as in *DP-FedAvg*, $\lambda_{\mathrm{p}}^* \to \frac{N}{\Gamma^2 N + \Gamma^2}$.
- $\alpha^2 \to 0$, i.e., no observation noise, $\lambda_{\mathrm{p}}^* \to 0$, and $\lambda_{\mathrm{np}}^* \to 0$. The optimal estimator approaches the local estimator, i.e., $\hat{\boldsymbol{\theta}}_j(\lambda_j^*) \to \hat{\boldsymbol{\phi}}_j$.
- $\tau^2 \to 0$, i.e., all clients have IID samples, $\lambda_{\mathrm{p}}^* \to \frac{N+\Gamma^2(N-N_{\mathrm{p}})}{\Gamma^2}$ and $\lambda_{\mathrm{np}}^* \to \infty$.

5.4.4 Optimality of *FeO2* in Linear Problems

Next, we show the convergence of the *FeO2* algorithm to the *FeO2* global and local objectives for the linear regression problem described above as follows

Lemma 5.4.5 (FeO2 convergence in linear problems) *FeO2, with learning rate $\eta = 1$ and $\eta_p = \frac{1}{1+\lambda_j}$, converges to the global FeO2 objective and the local FeO2 objective.*

Proof: In the considered setup, we denote $\hat{\boldsymbol{\phi}}_j = \frac{1}{n_s}\sum_{i=1}^{n_s} \boldsymbol{y}_{j,i}$ at client c_j. The client updates the global estimation $\boldsymbol{\theta}$ by minimizing the loss function in (5.4.5). The global estimation update at the client follows

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta(\boldsymbol{\theta} - \hat{\boldsymbol{\phi}}_j). \tag{5.4.40}$$

Updating the estimation once with $\eta = 1$ results in the global estimation update being $\hat{\boldsymbol{\phi}}_j$, adding the noise results in the same $\boldsymbol{\psi}_j$, and hence the global estimate in the next iteration is unchanged.

As for the local FeO2 estimation, when the client receives the global estimate $\boldsymbol{\theta}$ after the first round, the client updates its estimate $\boldsymbol{\theta}_j$ as

$$\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j - \eta_p\big((\boldsymbol{\theta}_j - \hat{\boldsymbol{\phi}}_j) + \lambda_j(\boldsymbol{\theta}_j - \boldsymbol{\theta})\big). \tag{5.4.41}$$

Updating the estimate once with $\eta_p = \frac{1}{1+\lambda_j}$ gives $\boldsymbol{\theta}_j = \frac{1}{1+\lambda_j}(\hat{\boldsymbol{\phi}}_j + \lambda_j\boldsymbol{\theta})$, which is the solution to the local FeO2 objective in (5.4.32). Hence, FeO2 converges to the global and local FeO2 objectives.

■

Next, we state the optimality theorem of *FeO2* algorithm for the considered setup described above.

Theorem 5.4.6 (FeO2 optimality in linear problems) *FeO2 from the server's point of view with ratio r^* chosen below, is Bayes optimal (i.e., $\boldsymbol{\theta}$ converges to $\boldsymbol{\theta}^*$) in the considered federated linear regression problem.*

$$r^* = \frac{\sigma_c^2}{\sigma_c^2 + N_p\gamma^2}. \tag{5.4.42}$$

Furthermore, FeO2 from the clients point of view, with λ_j^* chosen below, is Bayes optimal (i.e., $\boldsymbol{\theta}_j$ converges to $\boldsymbol{\theta}_j^*$ for each client $j \in [N]$) in the considered federated linear regression problem.

$$\lambda_j^* = \begin{cases} \frac{1}{\Upsilon^2}, & \textit{if } c_j \in \mathcal{C}_{np} \\ \frac{N+\Upsilon^2 N+\Gamma^2(N-N_p)}{\Upsilon^2(\Upsilon^2+1)N+\Upsilon^2\Gamma^2(N-N_p+1)+\Gamma^2}, & \textit{if } c_j \in \mathcal{C}_p \end{cases}. \tag{5.4.43}$$

Proof: This follows by observing Lemma 5.4.5, which states that the algorithm converges to the global and local FeO2 objectives, then by Lemma 5.4.2 and Lemma 5.4.4, which state that the

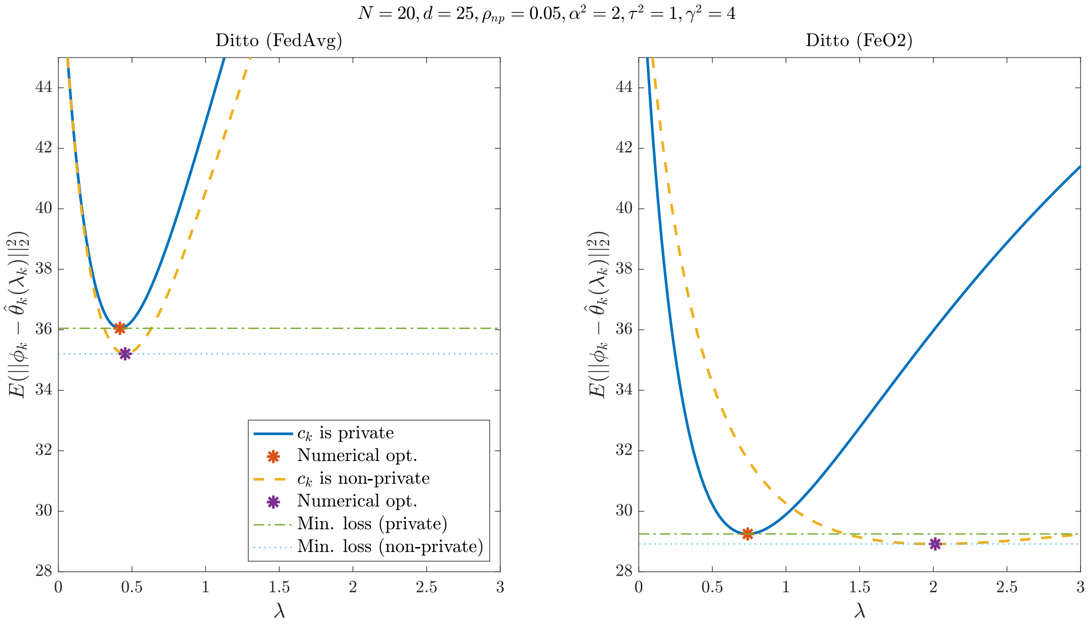


Figure 5.3: The effect of opting out on the personalized local model estimate for a linear regression problem as a function of λ when the server employs (left) vanilla *FedAvg* aggregation and (right) *FeO2* optimal aggregation.

solution to the FeO2 objective is the Bayes optimal solution for both global and local objectives.

■

5.4.5 Incentive For Opting Out Of DP

In this part, we would like to observe the effect of opting out of privacy on the client's personalized local model, compared to the one where the client remains private. We show an experiment comparing *Ditto* with *FeO2*, i.e., using r^*, against *Ditto* with *FedAvg*, i.e., $r = 1$ in Algorithm 5.1 for two scenarios. The first is when the client chooses to opt out of privacy, and the second is when the client chooses to remain private. See Figure 5.3 for the results of such experiment. We can see that the *Ditto* with *FeO2* outperforms the one with *FedAvg*, and the gain of opting out is clear in terms of reducing the loss at the client. Note that in Figure 5.3, *FeO2* reduces the minimum loss by around 18% compared to the *FedAvg* aggregator. We can see in the figure that when utilizing

Ditto with *FedAvg*, opting out reduces the minimum loss by about 2.5%, while in *Ditto* with *FeO2*, opting out reduces the minimum loss by 1.2%. It is worth noting that even for real-world federated datasets, such gain can be observed, as will be shown in the next section.

5.5 Experimental Results

In this section, we present the results of a number of experiments to show the gain in performance of the proposed *FeO2* algorithm compared to the baseline *DP-FedAvg* algorithm. The experiments show that *FeO2* outperforms *DP-FedAvg* with the right choice of the hyperparameter r in terms of the global model accuracy, as well as in terms of the average personalized local model accuracy.

5.5.1 Setup

The experiments are conducted on multiple federated datasets, synthetic and realistic. The synthetic datasets are manually created to simulate extreme cases of data heterogeneity often exhibited in federated learning scenarios. The realistic federated datasets are from Tensorflow Federated (TFF) [80], where such datasets are assigned to clients according to certain criteria. The synthetic dataset is referred to as the non-IID MNIST dataset, and the number of samples at a client is fixed across all clients. In this dataset, each client is assigned samples randomly from the subsets of samples each with a single digit between $0-9$. A skewed version of the synthetic dataset is one where non-private clients are sampled from the clients who only have the digit 7 in their data. In the non-IID MNIST dataset, we have $2,000$ clients and we randomly sample 5% of them for training each round. The realistic federated datasets are the federated MNIST (FMNIST) and the federated extended MNIST (FEMNIST) from TFF datasets. The FMNIST and FEMNIST datasets contain

$3,383$ and $3,400$ clients, respectively, and we sample around 3% of them for training each round. TensorFlow Privacy (TFP) [81] is used to compute the privacy loss incurred during training.

For all experiments, training is stopped after 500 communication rounds for each experiment. The server's test dataset is the test MNIST dataset in the non-IID MNIST experiments, or the collection of the test datasets of all clients in the FMNIST and FEMNIST datasets. Note that the experiment of *FeO2* with $r = 0$ denotes the case where the server only communicates with non-private clients during training and ignores all private clients. We vary the ratio hyperparameter r as well as the *Ditto* hyperparameters λ_{p} and λ_{np} and observe the results. The setup of each experiment is shown in Table 5.1, and the description of the models used in each experiment is shown in Table 5.2, while the hyperparameters used in each experiment are shown in Table 5.3. Note that the hyperparameters used in the non-IID MNIST datasets are the same for both the original dataset and the skewed dataset.

Table 5.1: Experiments setup: Number of clients is N, approximate fraction of clients per round is q_s.

Dataset	N	q_s	**Task**	**Model**
non-IID MNIST	2,000	5%	10-label classification	FC-NN
FMNIST	3,383	3%	10-label classification	FC-NN
FEMNIST	3,400	3%	62-label classification	CNN

5.5.2 Results

In this part, we provide the results of the experiments on the datasets mentioned above. In these experiments, we provide results for an opt-out rate of 5% of the total client population. Clients that opt out are picked randomly from the set of all clients but fixed across all experiments for a fair comparison. The exception for this assumption is for the skewed non-IID MNIST dataset, where

Table 5.2: Models used for experiments.

non-IID MNIST, and FMNIST Datasets		
Layer	**Size**	**Activation**
Input image	28×28	-
Flatten	784	-
Fully connected	50	ReLU
Fully connected	10	Softmax
FEMNIST Dataset		
Input image	28×28	-
Convolutional (2D)	$28 \times 28 \times 16$	ReLU
Max pooling (2D)	$14 \times 14 \times 16$	-
Convolutional (2D)	$14 \times 14 \times 32$	ReLU
Max pooling (2D)	$7 \times 7 \times 32$	-
Dropout (25%)	-	-
Flatten	1568	-
Fully connected	128	ReLU
Dropout (50%)	-	-
Fully connected	62	Softmax

Table 5.3: Hyperparameters used for each experiment.

Dataset Hyperparameter	non-IID MNIST	FMNIST	FEMNIST
Batch size	20		
Epochs	25	50	25
η, η_p	0.5	0.01	0.02
η, η_p decaying factor	0.9 every 50 rounds		N/A
S^0	0.5	0.5	2.0
η_b	0.2		
κ	0.5		
Effective noise multiplier	1.5	4.0	1.0

clients that opt out are sampled from the clients who have the digit 7. All other hyperparameters are fixed. To evaluate the performance of each algorithm, we measure the following quantities for each dataset:

1. Acc_{g}: the test accuracy on the *server's* test dataset using the global model.

2. $Acc_{\text{g,p}}$, $Acc_{\text{g,np}}$: the average test accuracy of all *private* and *non-private* clients using the global model on their local test datasets, respectively.

Table 5.4: Experiment results on *non-IID MNIST*, $(\epsilon, \delta) = (3.6, 10^{-4})$. The variance of the performance metric across clients is between parenthesis.

$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.005$								
Setup		Global model				Personalized local models		
Algorithm	**hyperparam.**	$Acc_g\%$	$Acc_{\text{g,p}}\%$	$Acc_{\text{g,np}}\%$	$\triangle_{\text{g}}\%$	$Acc_{\text{l,p}}\%$	$Acc_{\text{l,np}}\%$	$\triangle_{\text{l}}\%$
Non-private	-	93.8	-	93.75(0.13)	-	-	99.98(0.001)	-
DP-FedAvg	-	88.75	88.64(0.39)	-	-	99.97(0.002)	-	-
FeO2	$r=0$	90.7	90.64(0.68)	91.72(0.5)	1.08	90.64(0.68)	99.94(0.001)	9.2964
FeO2	$r=0.001$	91.74	91.65(0.39)	92.61(0.27)	0.94	99.94(0.001)	99.95(0.001)	0.0053
FeO2	$r=0.01$	92.48	92.43(0.30)	93.30(0.21)	0.88	99.94(0.001)	99.94(0.001)	-0.002
FeO2	$r=0.025$	92.36	92.28(0.27)	92.96(0.19)	0.68	99.95(0.001)	99.91(0.001)	-0.04
FeO2	$r=0.1$	90.7	90.59(0.34)	91.31(0.26)	0.73	99.97(0.001)	99.95(0.001)	-0.02
FeO2	$r=1$	87.71	87.55(0.42)	88.35(0.34)	0.8	99.97(0.001)	99.93(0.001)	-0.03
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.05$								
Non-private	-	93.81	-	93.76(0.13)	-	-	99.93(0.001)	-
DP-FedAvg	-	87.98	87.97(0.39)	-	-	99.84(0.002)	-	-
FeO2	$r=0$	91	91.12(0.48)	92.08(0.41)	0.96	91.12(0.48)	99.76(0.002)	8.65
FeO2	$r=0.001$	92.15	92.10(0.33)	92.88(0.25)	0.78	99.81(0.002)	99.78(0.002)	-0.03
FeO2	$r=0.01$	92.45	92.39(0.33)	93.26(0.25)	0.87	99.81(0.002)	99.78(0.003)	-0.03
FeO2	$r=0.025$	92.14	92.09(0.35)	93.01(0.26)	0.92	99.85(0.002)	99.8(0.002)	-0.05
FeO2	$r=0.1$	90.7	90.82(0.29)	91.55(0.21)	0.73	99.87(0.002)	99.80(0.003)	-0.06
FeO2	$r=1$	89.64	89.50(0.32)	90.55(0.24)	1.05	99.83(0.002)	99.84(0.002)	0.01
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.25$								
Non-private	-	93.79	-	93.75(0.13)	-	-	99.10(0.007)	-
DP-FedAvg	-	88.26	88.23(0.41)	-	-	98.23(0.017)	-	-
FeO2	$r=0$	90.42	90.41(0.69)	91.41(0.58)	1.0	90.41(0.69)	98.06(0.023)	7.65
FeO2	$r=0.001$	92.18	92.12(0.34)	92.85(0.26)	0.73	98.47(0.015)	98.08(0.024)	-0.39
FeO2	$r=0.01$	92.41	92.35(0.29)	93.19(0.21)	0.83	98.62(0.015)	98.33(0.019)	-0.28
FeO2	$r=0.025$	92.5	92.42(0.28)	93.19(0.19)	0.77	98.71(0.011)	98.41(0.017)	-0.3
FeO2	$r=0.1$	91.17	91.10(0.32)	91.94(0.24)	0.84	98.71(0.012)	98.60(0.013)	-0.11
FeO2	$r=1$	88.27	88.09(0.49)	89.08(0.4)	0.99	98.14(0.017)	98.13(0.017)	-0.01

3. $Acc_{\text{l,p}}$, $Acc_{\text{l,np}}$: the average test accuracy of all *private* and *non-private* clients using their personalized local models on their local test datasets, respectively.

4. $\triangle_{\text{g}}$, $\triangle_{\text{l}}$: the gain in the average performance of *non-private* clients over the *private* clients using the global model and the personalized local models on their local test datasets, respectively; computed as $\triangle_{\text{g}} = Acc_{\text{g,np}} - Acc_{\text{g,p}}$ and $\triangle_{\text{l}} = Acc_{\text{l,np}} - Acc_{\text{l,p}}$.

The results are shown in Tables 5.4-5.7, for each experiment along with their corresponding hyperparameters. For each experiment, the rows with the parameters that result in the best performance are highlighted. If two different sets of parameters result in two different competing results, such as one with a better global model performance at the server and one with better personalized local models at the clients, we highlight both.

Table 5.5: Experiment results on *Skewed non-IID MNIST*, $(\epsilon, \delta) = (3.6, 10^{-4})$. The variance of the performance metric across clients is between parenthesis.

$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.005$								
Setup		Global model				Personalized local models		
Algorithm	**hyperparam.**	$Acc_g\%$	$Acc_{\text{g,p}}\%$	$Acc_{\text{g,np}}\%$	$\triangle_{\text{g}}\%$	$Acc_{\text{l,p}}\%$	$Acc_{\text{l,np}}\%$	$\triangle_{\text{l}}\%$
Non-private	-	93.67	-	$93.62(0.15)$	-	-	$99.98(0.001)$	-
DP-FedAvg	-	88.93	$88.87(0.35)$	-	-	$99.98(0.001)$	-	-
FeO2	$r=0$	10.27	$7.1(6.5)$	$100(0)$	92.9	$7.1(6.5)$	$100(0)$	92.9
FeO2	$r=0.025$	87.11	$86.61(1.10)$	$98.16(0.01)$	11.55	$99.99(0.001)$	$99.91(0.001)$	-0.08
FeO2	$r=0.1$	90.36	$89.96(0.37)$	$97.45(0.01)$	7.49	$99.97(0.001)$	$99.76(0.003)$	-0.21
FeO2	$r=0.5$	88.44	$88.14(0.36)$	$93.36(0.03)$	5.2	$99.98(0.001)$	$99.93(0.001)$	-0.05
FeO2	$r=0.75$	89.14	$88.92(0.37)$	$92.43(0.06)$	3.5	$99.97(0.001)$	$99.93(0.001)$	-0.04
FeO2	$r=0.9$	87.96	$87.69(0.56)$	$92.97(0.04)$	5.28	$99.98(0.001)$	$99.96(0.001)$	-0.02
FeO2	$r=1$	88.25	$88.05(0.39)$	$89.98(0.05)$	1.93	$99.97(0.001)$	$99.85(0.001)$	-0.11
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.05$								
Non-private	-	93.67	-	$93.62(0.15)$	-	-	$99.93(0.001)$	-
DP-FedAvg	-	88.78	$88.70(0.53)$	-	-	$99.83(0.002)$	-	-
FeO2	$r=0$	10.28	$7.1(6.5)$	$100(0)$	92.9	$7.1(6.5)$	$100(0)$	92.9
FeO2	$r=0.025$	87.92	$87.45(0.99)$	$98.1(0.01)$	10.65	$99.95(0.001)$	$99.75(0.003)$	-0.2
FeO2	$r=0.1$	88.98	$88.64(0.52)$	$96.18(0.02)$	7.54	$99.9(0.001)$	$99.47(0.005)$	-0.43
FeO2	$r=0.5$	88.22	$87.9(0.38)$	$93.43(0.03)$	5.33	$99.85(0.002)$	$99.42(0.008)$	-0.42
FeO2	$r=0.75$	88.56	$88.37(0.35)$	$91.33(0.04)$	2.94	$99.84(0.002)$	$99.52(0.004)$	-0.33
FeO2	$r=0.9$	89.19	$88.97(0.4)$	$92.24(0.03)$	3.27	$99.88(0.001)$	$99.58(0.005)$	-0.3
FeO2	$r=1$	88.33	$88.11(0.46)$	$91.67(0.04)$	3.56	$99.87(0.001)$	$99.61(0.001)$	-0.26
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.25$								
Non-private	-	93.67	-	$93.62(0.15)$	-	-	$99.09(0.007)$	-
DP-FedAvg	-	87.78	$87.71(0.53)$	-	-	$98.15(0.02)$	-	-
FeO2	$r=0$	10.27	$7.1(6.5)$	$100(0)$	92.9	$7.1(6.5)$	$100(0)$	92.9
FeO2	$r=0.025$	87.51	$87.01(0.9)$	$98.49(0.01)$	11.48	$98.69(0.01)$	$99.09(0.006)$	-0.4
FeO2	$r=0.1$	89.05	$88.66(0.54)$	$96.8(0.02)$	8.14	$98.69(0.012)$	$98.55(0.008)$	-0.13
FeO2	$r=0.5$	88.18	$88.11(0.55)$	$93.43(0.03)$	5.32	$98.32(0.014)$	$97.80(0.01)$	-0.52
FeO2	$r=0.75$	87.96	$87.8(0.33)$	$92.58(0.03)$	4.78	$98.25(0.017)$	$97.5(0.02)$	-0.75
FeO2	$r=0.9$	88.26	$87.93(0.41)$	$91.67(0.03)$	3.74	$98.25(0.02)$	$97.68(0.02)$	-0.57
FeO2	$r=1$	89.4	$89.22(0.26)$	$92.01(0.03)$	2.79	$98.27(0.02)$	$97.62(0.03)$	-0.64

We can see from Tables 5.4-5.7 that *FeO2* enables the server to learn better global models, as well as clients to learn better personalized local models compared to the baseline private FL, i.e., *DP-FedAvg*. For example, the gain due to *FeO2* compared to the *DP-FedAvg* in terms of global model performance is 3.8% on average and is up to 9.27%. For personalized local models, the gain in the average accuracy for clients due to *FeO2* compared to *DP-FedAvg* is up to 9.99%. The average gap in the average performance of personalized local models between *DP-FedAvg* and *non-private* is 3.57%, which is reduced to 0.95% between *FeO2* and *non-private*. Additionally, we can also see the gain in the average performance in personalized local models between clients who choose to opt out of privacy and clients who choose to remain private. This demonstrates the advantage of opting out of privacy, which provides clients with an incentive to opt out of

Table 5.6: Experiment results on *FMNIST*, $(\epsilon, \delta) = (0.6, 10^{-4})$. The variance of the performance metric across clients is between parenthesis.

$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.005$								
Setup		Global model				Personalized local models		
Algorithm	**hyperparam.**	$Acc_g\%$	$Acc_{\text{g,p}}\%$	$Acc_{\text{g,np}}\%$	$\triangle_{\text{g}}\%$	$Acc_{\text{l,p}}\%$	$Acc_{\text{l,np}}\%$	$\triangle_{\text{l}}\%$
Non-private	-	89.65	-	$89.35(1.68)$	-	-	$93.95(0.67)$	-
DP-FedAvg	-	71.76	$71.42(2.79)$	-	-	$91.01(0.94)$	-	-
FeO2	$r=0$	81.78	$80.73(2.45)$	$89.35(1.5)$	8.62	$80.73(2.4)$	$95.80(0.39)$	15.06
FeO2	$r=0.01$	85.38	$84.61(2.05)$	$89.3(1.26)$	4.69	$93.26(0.74)$	$95.94(0.41)$	2.67
FeO2	$r=0.025$	85.7	$84.93(1.97)$	$89.58(1.29)$	4.65	$93.04(0.76)$	$95.22(0.54)$	2.18
FeO2	$r=0.05$	85.21	$84.68(1.99)$	$86.22(1.76)$	1.54	$92.87(0.74)$	$95.40(0.51)$	2.53
FeO2	$r=0.1$	81.76	$81.45(2.45)$	$81.96(1.84)$	0.51	$92.47(0.78)$	$94.83(0.52)$	2.36
FeO2	$r=0.5$	78.19	$78.02(2.59)$	$76.48(3.02)$	-1.53	$91.08(0.94)$	$92.59(0.83)$	1.51
FeO2	$r=1$	75.87	$75.77(2.84)$	$74.41(2.8)$	-1.36	$90.45(1.02)$	$92.32(0.8)$	1.87
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.05$								
Non-private	-	89.65	-	$89.35(1.68)$	-	-	$94.53(0.59)$	-
DP-FedAvg	-	77.61	$77.62(2.55)$	-	-	$90.04(1.04)$	-	-
FeO2	$r=0$	82.61	$80.72(2.45)$	$89.45(1.51)$	8.73	$80.72(2.45)$	$95.57(0.38)$	14.84
FeO2	$r=0.01$	86.88	$85.36(1.89)$	$90.02(1.28)$	4.66	$93.76(0.68)$	$95.78(0.36)$	2.02
FeO2	$r=0.025$	86.03	$84.22(1.98)$	$88.40(1.68)$	4.18	$93.53(0.68)$	$95.11(0.54)$	0.52
FeO2	$r=0.05$	84.65	$82.68(2.16)$	$86.68(1.67)$	4.00	$92.92(0.76)$	$95.02(0.55)$	2.1
FeO2	$r=0.1$	82.89	$81.72(2.28)$	$83.68(2.18)$	1.96	$92.38(0.83)$	$94.25(0.61)$	1.87
FeO2	$r=0.5$	76.59	$78.05(2.60)$	$78.04(2.66)$	-0.0041	$89.63(1.10)$	$91.67(0.84)$	2.04
FeO2	$r=1$	72.42	$77.14(2.72)$	$76.28(2.76)$	-0.86	$89.12(1.15)$	$90.92(0.91)$	1.8
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.25$								
Non-private	-	89.66	-	$89.36(1.69)$	-	-	$94.32(0.64)$	-
DP-FedAvg	-	70.1	$70.40(2.91)$	-	-	$88.38(1.25)$	-	-
FeO2	$r=0$	81.93	$80.85(2.39)$	$89.71(1.39)$	8.86	$80.85(2.39)$	$94.56(0.50)$	13.71
FeO2	$r=0.01$	85.31	$84.55(1.98)$	$89.27(1.54)$	4.72	$92.76(0.78)$	$94.77(0.5)$	2.01
FeO2	$r=0.025$	86.17	$85.52(1.92)$	$89.25(1.31)$	3.73	$92.46(0.85)$	$94.35(0.57)$	1.89
FeO2	$r=0.05$	83.97	$83.5(2.19)$	$85.4(1.88)$	1.9	$91.69(0.91)$	$93.9(0.53)$	2.21
FeO2	$r=0.1$	83.78	$83.22(2.11)$	$84.94(2.12)$	1.72	$90.9(1.02)$	$92.62(0.73)$	1.72
FeO2	$r=0.5$	74.64	$74.63(3.23)$	$72.54(2.93)$	-2.09	$88.12(1.34)$	$88.69(1.28)$	0.57
FeO2	$r=1$	72.67	$72.05(2.83)$	$74.31(2.42)$	2.26	$87.54(1.34)$	$87.39(1.23)$	-0.15

differential privacy if they look to improve their personalized local models. For example, non-private clients can gain up to 3.49% on average in terms of personalized local model performance compared to private clients. It is worth mentioning that opting out can also improve the global model's performance on clients' local data. We observe that there is up to 12.4% gain in the average performance of non-private clients in terms of the accuracy of the global model on the local data compared to the one of baseline *DP-FedAvg*.

Table 5.7: Experiment results on *FEMNIST*, $(\epsilon, \delta) = (4.1, 10^{-4})$. The variance of the performance metric across clients is between parenthesis.

$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.005$								
Setup		Global model				Personalized local models		
Algorithm	**hyperparam.**	$Acc_g\%$	$Acc_{\text{g,p}}\%$	$Acc_{\text{g,np}}\%$	$\triangle_{\text{g}}\%$	$Acc_{\text{l,p}}\%$	$Acc_{\text{l,np}}\%$	$\triangle_{\text{l}}\%$
Non-private	-	81.56	-	$81.72(1.37)$	-	-	$73.86(1.5)$	-
DP-FedAvg	-	75.39	$76.1(1.73)$	-	-	$71.3(1.47)$	-	-
FeO2	$r=0$	72.77	$75.34(2.6)$	$85.7(1.14)$	10.36	$75.34(2.6)$	$72.2(1.22)$	-3.14
FeO2	$r=0.001$	73.66	$76.22(2.44)$	$86.04(1.2)$	9.82	$72.78(1.51)$	$71.74(1.34)$	-1.04
FeO2	$r=0.01$	74.75	$77.16(2.26)$	$86.4(1.07)$	9.24	$73.24(1.52)$	$71.49(1.29)$	-1.76
FeO2	$r=0.025$	75.37	$77.66(2.06)$	$86.56(1)$	8.9	$73.11(1.55)$	$71.62(1.18)$	-1.49
FeO2	$r=0.1$	76.47	$77.99(1.68)$	$84.36(1.31)$	6.37	$72.3(1.5)$	$69.93(1.15)$	-2.37
FeO2	$r=0.5$	76.11	$76.69(1.62)$	$80.82(1.3)$	4.13	$71.32(1.56)$	$70.11(1.26)$	-1.2
FeO2	$r=1$	74.96	$75.6(1.69)$	$77.84(1.47)$	2.24	$71.44(1.5)$	$70.03(1.18)$	-1.4
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.05$								
Non-private	-	81.95	-	$82.09(1.38)$	-	-	$82.89(1.13)$	-
DP-FedAvg	-	75.42	$75.86(1.82)$	-	-	$74.69(1.29)$	-	-
FeO2	$r=0$	72.65	$75.9(2.5)$	$86.19(1.27)$	10.29	$80.59(1.13)$	$81.97(0.88)$	1.38
FeO2	$r=0.001$	73.31	$75.9(2.5)$	$86.19(1.27)$	10.29	$80.59(1.13)$	$81.97(0.88)$	1.38
FeO2	$r=0.01$	74.68	$77.16(2.27)$	$86.25(1.05)$	9.09	$80.74(1.06)$	$82.13(0.98)$	1.38
FeO2	$r=0.025$	75.22	$77.43(2.09)$	$85.95(1.12)$	8.52	$80(1.16)$	$80.99(0.92)$	1.01
FeO2	$r=0.1$	76.52	$77.91(1.67)$	$83.9(1.27)$	5.99	$77.9(1.22)$	$79.15(0.99)$	1.25
FeO2	$r=0.5$	76.15	$76.55(1.68)$	$80.04(1.62)$	3.49	$75.43(1.25)$	$77.13(1.17)$	1.7
FeO2	$r=1$	75.12	$75.87(1.65)$	$78.59(1.58)$	2.72	$74.67(1.34)$	$75.95(1.12)$	1.28
$\lambda_{\text{p}} = \lambda_{\text{np}} = 0.25$								
Non-private	-	81.66	-	$81.79(1.38)$	-	-	$84.46(0.89)$	-
DP-FedAvg	-	75.99	$76.56(1.6)$	-	-	$73.06(1.46)$	-	-
FeO2	$r=0$	72.89	$75.5(2.56)$	$86.09(1.28)$	10.6	$75.5(2.56)$	$84.77(0.8)$	9.28
FeO2	$r=0.001$	73.41	$76.01(2.51)$	$85.99(1.13)$	9.97	$80.98(1.06)$	$84.71(0.83)$	3.73
FeO2	$r=0.01$	74.86	$77.31(2.18)$	$86.73(0.98)$	9.42	$81.19(1.02)$	$84.68(0.78)$	3.49
FeO2	$r=0.025$	75.41	$77.68(2.1)$	$86.23(1.03)$	8.55	$80.01(1.1)$	$83.2(0.8)$	3.19
FeO2	$r=0.1$	76.62	$77.82(1.68)$	$83.35(1.27)$	5.52	$76.99(1.24)$	$78.96(1.04)$	1.97
FeO2	$r=0.5$	75.89	$76.71(1.65)$	$80.01(1.4)$	3.3	$73.48(1.37)$	$75.49(1.57)$	2.01
FeO2	$r=1$	75.31	$75.67(1.71)$	$78.88(1.59)$	3.21	$72.58(1.45)$	$74.98(1.43)$	2.4

5.6 Conclusion

In this chapter, a new aspect of heterogeneity in federated learning setups is considered, namely heterogeneity in privacy requirements. A new setup is introduced for privacy heterogeneity between clients where privacy is no longer necessary for all clients, and some clients choose to opt out of privacy. A new algorithm called *FeO2* for the considered setup is proposed. In *FeO2*, the aim is to employ differential privacy to maintain the privacy of clients who choose to remain private and utilize the additional information from the non-private clients to improve the utility of the model. Additionally, *Ditto* is utilized as a personalization scheme to examine whether *FeO2* enhances the performance for personalized FL. An analytical treatment for the federated linear regression problem and showed the optimality of *FeO2* from the server's point of view as well as the clients' point of view, when combined with *Ditto*. Specifically, the optimal value of the ratio hyperparameter at the server and the optimal values of the *Ditto* parameter at clients that achieve the best performance were computed. Finally, a set of experiments on synthetic and realistic federated datasets were conducted and showed that *FeO2* outperforms the baseline private FL algorithm in terms of the global model as well as the personalized local models' performance, and showed the incentive of becoming non-private compared to remaining private in such scenarios in terms of the gain in the average performance.

CHAPTER 6

Conclusion and Future Work

The unprecedented growth of distributed devices, along with the incredible amount of sensitive data they generate, dictated a set of new challenges to existing algorithms, especially the ones that are concerned with the security and privacy of data. One of the main challenges is the limited resources available to such devices, such as limited computational capabilities, batteries, etc. Therefore, any algorithm needs to be efficient to be implemented on such devices. In Chapter 2, low-complexity protocols for generating secret keys for devices operating in static environments are presented, where legitimate parties generate local randomness and exchange it to improve the key generation rate. The reliability and security of the protocols are characterized for the proposed protocols by upper-bounding the probability of accepting a mismatched key and the probability of a successful eavesdropping attack, respectively. Additionally, simulations are provided to evaluate the performance of the protocols using secret key generation metrics.

The notion of threshold-security is presented in Chapter 3, where the security of the entire message is not required, but rather the security of all sub-blocks of the message with size up to a certain threshold is desired. The formulation of such a notion of security using information-theoretic measures is provided, along with proposing a general coding scheme based on linear block codes. A low-complexity threshold-secure coding scheme based on Reed-Muller codes for noiseless channels is presented along with its successive cancellation decoder. Furthermore, a low-complexity threshold-secure robust coding scheme based Reed-Muller codes for the binary erasure main channel, as well as its decoder, are presented.

In Chapter 4, perfect machine unlearning for regression problems is considered. Designed for ensemble learning setups, an approach utilizing the encoding of training samples using random linear encoders is presented. This approach enables us to encode the training dataset to a smaller dataset where the performance of the resulting model outperforms the one of the baseline uncoded unlearning algorithm in terms of performance vs unlearning cost. Experimental evaluation of the algorithm is presented to compare its performance to the baseline along with a discussion on the results.

A new setup for privacy-preserving federated learning is presented in Chapter 5. Privacy heterogeneity is proposed for federated learning setups where clients are no longer mandated to be private, but rather assumed to be private and given the choice to opt out of privacy. A new algorithm is presented to take advantage of the newly introduced setup that maintains the privacy of clients in the set of private clients and utilizes the additional information from the set of non-private clients to improve the utility of the model. The algorithm is analyzed in the simplified setup of federated linear regression and its optimality is shown. Moreover, The proposed algorithm is compared to the baseline private federated learning algorithm using several synthetic and realistic datasets, where it is observed that the proposed algorithm outperforms the baseline in terms of the performance of the global server's model as well as clients' local models.

Future Work

Some interesting directions can be further explored for the secret key generation setups. For example, secret key generation protocols are presented for setups where two parties need a shared key. However, as the number of small devices operating in such environments increases, there may be a need for multiple devices to generate a secret key that is shared between them to be used for broadcast messages. Additionally, investigating scenarios where the single passive eavesdropper has additional capabilities, e.g., employing multiple eavesdroppers or antennas, employing active jamming to disrupt the key generation process, etc., is also another direction of future work.

Furthermore, efficient threshold-secure coding schemes were presented based on Reed-Muller codes, which limits the possible code rates, i.e., key and message length. One can further examine whether a solution based on punctured Reed-Muller codes can be utilized for threshold security, where ideas from polar codes can be useful. Furthermore, studying additional setups, such as wiretap channels, with the aim of achieving threshold security is another interesting direction.

Additionally, the coded machine unlearning algorithm proposes a solution for regression problems. Extending the proposed algorithm to different classes of learning models such as logistic regression, deep neural networks, etc., as well as utilizing different classes of codes other than linear random codes is an interesting direction of future work. Moreover, theoretical examination of the role of coding in machine unlearning and the interplay between influential samples with random coding, and their impact on the learned model is another possible direction of further exploration.

There are multiple directions of future work for the federated learning with opt-out differential privacy setup. For example, further examination of the algorithm to include additional experiments on more complex datasets as well as learning models. Additionally, the proposed setup only considers two choices of privacy; however, one can further examine various levels of privacy where clients can choose the desired values of the privacy parameters rather than the proposed opt-in opt-out setup. Consequently, modifications to the algorithm need to be done to accommodate this level of privacy granularity for central differential privacy, or the introduction of local differential privacy needs to be implemented at clients rather than the server.

BIBLIOGRAPHY

[1] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021.

[3] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[4] CE Rasmussen, RM Neal, G Hinton, D Camp, M Revow, Z Ghahramani, R Kustra, and R Tibshirani. Data for evaluating learning in valid experiments (delve), 2003.

[5] IDC. Data volume of internet of things (IoT) connections worldwide in 2019 and 2025 (in zettabytes), Jul 2020.

[6] IoT Analytics. Internet of things (IoT) and non-IoT active device connections worldwide from 2010 to 2025 (in billions), Nov 2020.

[7] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[8] Aaron D Wyner. The wire-tap channel. *The bell system technical journal*, 54(8):1355–1387, 1975.

[9] Hessam Mahdavifar and Alexander Vardy. Achieving the secrecy capacity of wiretap channels using polar codes. *IEEE Transactions on Information Theory*, 57(10):6428–6443, 2011.

[10] Demijan Klinc, Jeongseok Ha, Steven W McLaughlin, Joao Barros, and Byung-Jae Kwak. Ldpc codes for the gaussian wiretap channel. *IEEE Transactions on Information Forensics and Security*, 6(3):532–540, 2011.

[11] Ling Liu, Yanfei Yan, and Cong Ling. Achieving secrecy capacity of the gaussian wiretap channel with polar lattices. *IEEE Transactions on Information Theory*, 64(3):1647–1665, 2018.

[12] Ueli M Maurer. Secret key agreement by public discussion from common information. *IEEE transactions on information theory*, 39(3):733–742, 1993.

[13] Rudolf Ahlswede and Imre Csiszár. Common randomness in information theory and cryptography. i. secret sharing. *IEEE Transactions on Information Theory*, 39(4):1121–1132, 1993.

[14] NIST FIPS Pub. 197: Advanced encryption standard (AES). *Federal information processing standards publication*, 197(441):0311, 2001.

[15] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[16] Eduard A Jorswieck, Anne Wolf, and Sabrina Engelmann. Secret key generation from reciprocal spatially correlated MIMO channels. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 1245–1250. IEEE, 2013.

[17] Masoud Ghoreishi Madiseh, Stephen W Neville, and Michael L McGuire. Applying beamforming to address temporal correlation in wireless channel characterization-based secret key generation. *IEEE Transactions on Information Forensics and Security*, 7(4):1278–1287, 2012.

[18] Shyamnath Gollakota and Dina Katabi. Physical layer wireless security made fast and channel independent. In *INFOCOM, 2011 Proceedings IEEE*, pages 1125–1133. IEEE, 2011.

[19] Satashu Goel and Rohit Negi. Guaranteeing secrecy using artificial noise. *IEEE transactions on wireless communications*, 7(6), 2008.

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

[21] Wei Zhang, Chongfu Zhang, Wei Jin, Chen Chen, Ning Jiang, and Kun Qiu. Chaos coding-based QAM IQ-encryption for improved security in OFDMA-PON. *IEEE Photonics Technology Letters*, 26(19):1964–1967, 2014.

[22] Fei Huo and Guang Gong. A new efficient physical layer OFDM encryption scheme. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1024–1032. IEEE, 2014.

[23] Junqing Zhang, Alan Marshall, Roger Woods, and Trung Q Duong. Design of an OFDM physical layer encryption scheme. *IEEE Transactions on Vehicular Technology*, 66(3):2114–2127, 2017.

[24] M Tahir, Sigit PW Jarot, and MU Siddiqi. Wireless physical layer security using encryption and channel pre-compensation. In *2010 International Conference on Computer Applications and Industrial Electronics*, pages 304–309. IEEE, 2010.

[25] Andre Zuquete and Joao Barros. Physical-layer encryption with stream ciphers. In *2008 IEEE International Symposium on Information Theory*, pages 106–110. IEEE, 2008.

[26] Young-Sik Kim, Jong-Hwan Kim, and Sang-Hyo Kim. A secure information transmission scheme with a secret key based on polar coding. *IEEE Communications Letters*, 18(6):937–940, 2014.

[27] Ning Cai and Raymond W Yeung. Secure network coding. In *Proceedings IEEE International Symposium on Information Theory,*, page 323. IEEE, 2002.

[28] Kapil Bhattad, Krishna R Narayanan, et al. Weakly secure network coding. In *Proceedings First Workshop on Network Coding, Theory, and Applications (NetCod)*, 2005.

[29] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

[30] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. *ICML*, 2020.

[31] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, pages 3513–3526, 2019.

[32] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020.

[33] 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.

[34] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[35] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[36] Lichao Sun, Jianwei Qian, Xun Chen, and Philip S Yu. LDP-FL: Practical private aggregation in federated learning with local differential privacy. *arXiv preprint arXiv:2007.15789*, 2020.

[37] Muah Kim, Onur Günlü, and Rafael F Schaefer. Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2650–2654. IEEE, 2021.

[38] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[39] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[40] Galen Andrew, Om Thakkar, H Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[41] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[42] Nasser Aldaghri and Hessam Mahdavifar. Fast secret key generation in static environments using induced randomness. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[43] Nasser Aldaghri and Hessam Mahdavifar. Physical layer secret key generation in static environments. *IEEE Transactions on Information Forensics and Security*, 15:2692–2705, 2020.

[44] Nasser Aldaghri and Hessam Mahdavifar. Threshold-secure coding with shared key. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 552–559. IEEE, 2019.

[45] Nasser Aldaghri and Hessam Mahdavifar. Threshold-secure coding with shared key. *IEEE Journal on Selected Areas in Information Theory*, 2(1):95–105, 2021.

[46] Nasser Aldaghri, Hessam Mahdavifar, and Ahmad Beirami. Coded machine unlearning. *IEEE Access*, 2021.

[47] Nasser Aldaghri, Hessam Mahdavifar, and Ahmad Beirami. FeO2: Federated learning with opt-out differential privacy. *arXiv preprint arXiv:2110.15252*, 2021.

[48] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36. ACM, 1999.

[49] A Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[50] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *International conference on the theory and applications of cryptographic techniques*, pages 523–540. Springer, 2004.

[51] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[52] Wei Xi, Xiang-Yang Li, Chen Qian, Jinsong Han, Shaojie Tang, Jizhong Zhao, and Kun Zhao. KEEP: Fast secret key extraction protocol for D2D communication. In *Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of*, pages 350–359. IEEE, 2014.

[53] Andreas F Molisch. *Wireless communications*, volume 34. John Wiley & Sons, 2012.

[54] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[55] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In *Advances in Cryptology–CRYPTO 2012*, pages 294–311. Springer, 2012.

[56] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[57] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, and Elaine Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-Allen and Hamilton Inc Mclean Va, 2001.

[58] Shu Sun, George R MacCartney, and Theodore S Rappaport. A novel millimeter-wave channel simulator and applications for 5G wireless communications. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2017.

[59] Pengfei Huang and Xudong Wang. Fast secret key generation in static wireless networks: A virtual channel approach. In *2013 Proceedings IEEE INFOCOM*, pages 2292–2300. IEEE, 2013.

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

[61] René Guillaume, Stephan Ludwig, Andreas Müller, and Andreas Czylwik. Secret key generation from static channels with untrusted relays. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*, pages 635–642. IEEE, 2015.

[62] Song Fang, Ian Markwood, and Yao Liu. Manipulatable wireless key establishment. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.

[63] Kai Zeng. Physical layer key generation in wireless networks: challenges and opportunities. *IEEE Communications Magazine*, 53(6):33–39, 2015.

[64] Serguei Primak, Kang Liu, and Xianbin Wang. Secret key generation using physical channels with imperfect CSI. In *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, pages 1–5. IEEE, 2014.

[65] Yagiz Sutcu, Husrev Taha Sencar, and Nasir Memon. A secure biometric authentication scheme based on robust hashing. In *Proceedings of the 7th workshop on Multimedia and security*, pages 111–116, 2005.

[66] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[67] Ron Roth. *Introduction to coding theory*. Cambridge University Press, 2006.

[68] Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, 2009.

[69] Hessam Mahdavifar, Mostafa El-Khamy, Jungwon Lee, and Inyup Kang. Fast multi-dimensional polar encoding and decoding. In *2014 Information Theory and Applications Workshop (ITA)*, pages 1–5. IEEE, 2014.

[70] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* Adaptive Computation and Machine Learning series, 2018.

[71] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

[72] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[73] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[74] David L Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[75] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[76] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*, volume 571. John Wiley & Sons, 2005.

[77] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[78] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. *International Conference on Machine Learning*, 2021.

[79] Hessam Mahdavifar, Ahmad Beirami, Behrouz Touri, and Jeff S Shamma. Global games with noisy information sharing. *IEEE Transactions on Signal and Information Processing over Networks*, 4(3):497–509, 2018.

[80] Google. TensorFlow Federated, 2019.

[81] Google. TensorFlow Privacy, 2018.