ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Privacy-Preserving Federated Learning for Cyber Threat Intelligence Sharing

Master Thesis

Benjamin Fischer

March 30, 2023

Advisors: Prof. Dr. Kenny Paterson, Dr. Juan R. Troncoso-Pastoriza

Applied Cryptography Group
Institute of Information Security
Department of Computer Science, ETH Zürich

Abstract

Collective cyber defense leveraging *Cyber Threat Intelligence* (CTI) sharing has recently gained considerable interest in addressing the rise of cyber threats and incidents that organizations and individuals face by training advanced machine learning models to predict and detect future threats. However, cyber threat intelligence sharing remains suboptimal due to the lack of confidentiality guarantee for shared cyber threat information, holding back stakeholders from contributing to collective cyber security.

Federated Learning (FL) is a decentralized approach to machine learning that enables multiple parties to train models collaboratively by keeping their datasets locally while only exchanging model updates. Recent work has revealed that federated learning is vulnerable to various inference attacks by analyzing the leakage of released local and global models.

In this work, we propose a hybrid *Privacy-Preserving Federated Learning* (PPFL) construction to address the problem of cyber threat intelligence sharing in an K-party federated setting. Our construction combines *Multi-Party Homomorphic Encryption* (MHE) and *Differential Privacy* (DP) to guarantee the confidentiality of the training data and intermediate models in the passive-adversary threat model, assuming collusions up to $K - 1$ parties.

Our experimental results show that our hybrid solution outperforms fully encrypted learning and achieves similar accuracy to non-private federated learning. We evaluate our hybrid construction by training neural networks on the MNIST and CIFAR-10 datasets, distributed among 3 parties.

Acknowledgement

First and foremost, I express my heartfelt gratitude to my supervisor, Dr. Juan R. Troncoso-Pastoriza, for his guidance, patience, and support throughout this research. His expertise and experience have contributed considerably to shaping the outcome of this thesis.

I deeply thank Prof. Dr. Kenny Paterson for allowing me to present this thesis to the Applied Cryptography Group. His outstanding course on applied cryptography strongly reinforced my interest in the subject.

A special thanks to Thierry Bossy and Jean-Philippe Bossuat for their valuable advice and our insightful discussions on deep learning and homomorphic encryption. Their contribution have been significant in improving the quality of this research.

Finally, I sincerely thank the entire team at Tune Insight, who has fostered an excellent working environment that has allowed me to grow and develop personally and professionally. Your encouragement and continued support have been a constant source of motivation.

I am truly grateful for all the support I have received and for making this research journey memorable.

Contents

List of Acronyms

CTI	Cyber Threat Intelligence
PET	Privacy-Enhancing Technology
SMC	Secure Multi-Party Computation
HE	Homomorphic Encryption
MHE	Multi-Party Homomorphic Encryption
DP	Differential Privacy
LDP	Local Differential Privacy
GDP	Global Differential Privacy
ML	Machine Learning
FL	Federated Learning
PPML	Privacy-Preserving Machine Learning
PPFL	Privacy-Preserving Federated Learning
NN	Neural Network
SGD	Stochastic Gradient Descent

Notations

Table 1. summarizes the common mathematical notations used in this work.

Symbol	Description
K	Total number of parties
P_i	i^{th} Party
D	Dataset or Database
x	Data record
n	Number of samples
m	Number of features
θ	Model parameters
ℓ_i	i^{th} Layer of θ
d	Model dimension
B	Mini-batch
η	Learning rate
C	Clipping constant
$\mathcal{L}(\theta)$	Loss function
$\nabla_\theta \mathcal{L}(\theta, x)$	Gradient of the loss function
R	Total number of rounds (global iterations)
T	Total number of local iterations
l	Total number of layers
h_j	Number of neurons in the j^{th} layer
X_i	Input matrix of P_i
W_j^t	Weight matrix for layer j at t^{th} iteration
A_j^t	Activation matrix for layer j at t^{th} iteration
$\phi(\cdot)$	Activation function
$\phi'(\cdot)$	Derivative of the activation function
E_j^t	Error matrix propagated in layer j at t^{th} iteration
∇W_j^t	Gradient matrix for layer j at t^{th} iteration
$\times$	Matrix multiplication
$\odot$	Hadamard product (element-wise multiplication)
λ	Security level
q_i	i^{th} Moduli
L	Initial level of a fresh ciphertext
S	Initial scale (bit precision) of a fresh ciphertext
Q_L	Ring modulus at initial level L
N	Ring dimension
$\mathbb{Z}_{Q_L}[X]$	Polynomial ring
$\mathcal{R}_{Q_L}$	Quotient ring
sk_i	Secret key of P_i
pk	Collective public key
msg	Arbitrary message

$\bar{p}$	Plaintext (encoded message msg)
$\boldsymbol{c}_{pk}$	Ciphertext
$L_{\boldsymbol{c}}$	Current level of ciphertext $\boldsymbol{c}$
$S_{\boldsymbol{c}}$	Current scale (bit precision) of ciphertext $\boldsymbol{c}$
$\prod$	K-party distributed protocol

Table 1: Frequently Used Symbols and Notations.

Chapter 1

Introduction

In this chapter, we first justify the need for global *Cyber Threat Intelligence* (CTI) sharing to address current cyber defense challenges and then introduce *Privacy-Preserving Federated Learning* (PPFL) to enable such decentralized data-sharing scenarios. We then review related and concurrent PPFL constructions, and list the main contributions of our work.

1.1 Overview & Motivation

1.1.1 Cyber Threat Intelligence Sharing

Cyber Defense Challenges

Over the past two decades, the dramatic increase in new cyber threats and incidents has become a growing concern for individuals and organizations. These incidents, which may result in data breaches, system downtime, or intellectual property theft, can be devastating, as they usually induce reputation damage, business disruption, and financial losses.

As attackers constantly devise new ways to circumvent security systems, companies must stay on top of the latest threats to keep up with the ever-changing nature of modern cyber attacks and proactively adjust their detection and mitigation systems accordingly. Incident response time and coordination are critical to defend against these threats. As a consequence, organizations and institutions must allocate a significant budget to cyber defense to mitigate these new threats and improve their cyber resilience.

Collaborative Cyber Security

Access to real-time threat intelligence data is also crucial in effectively addressing the growing number of cyber incidents organizations face. Indeed, companies can improve their defense capabilities by globally sharing their

cyber information, such as malware hashes, system logs, or source IP addresses of phishing attempts, allowing security analysts to train advanced models to predict and detect future incidents. Furthermore, sharing cyber information can benefit the offensive side by enabling joint investigations to combat cyber crime. For these reasons, collective cyber defense is therefore gaining interest in the cyber security industry.

Cyber Information Retention

Implementing CTI sharing faces the *Free-Rider Problem*, describing the situation in which all participants want to reap the benefits of collective defense and improved threat response but restrain from sharing sensitive cyber information due to the lack of confidentiality guarantees. As a result, actors tend to limit the information they share, reducing the effectiveness of collaborative cyber security.

This problem expresses the trade-off between the benefits of improved threat response capabilities and the drawbacks of disclosing sensitive cyber information. CTI sharing thus remains sub-optimal, preventing the full potential of collective defense from being realized. It is therefore essential to implement robust privacy protection mechanisms to guarantee the confidentiality of shared cyber intelligence and encourage stakeholders to contribute to collaborative cyber security.

Privacy-Preserving Information Sharing

To this end, the 2020 annual report of the *World Economic Forum* (WEF) [3] highlights the combination of *Machine Learning* (ML) and *Privacy-Enhancing Technologies* (PETs), as a promising privacy-preserving framework for information sharing that can address the current cyber defense challenges [45]. On the one hand, ML algorithms enable automating cyber attack diagnosis. On the other hand, PETs provide security guarantees for data analysis while preserving utility.

This new information-sharing paradigm allows institutions to minimize the risk of exposure of their sensitive cyber information while benefiting from improved predictive and preventive defenses. In terms of implementation, CTI sharing is one of the many use-cases of PPFL that we introduce below.

1.1.2 Privacy-Preserving Federated Learning

Federated Learning (FL) is an ML technique that collaboratively trains a global model on multiple datasets distributed over separate clients without any data exchange, ensuring some level of data privacy by design.

The rapid development of FL is mainly due to the massive success of ML applications with the explosion of big data as well as legal regulations on

data privacy protection worldwide, such as the *General Data Protection Regulation* [19] in the EU, the *California Privacy Rights Act* [7] in the US, and the *Personal Data Protection Act* [24] in Singapore. These regulations emerging from user data privacy concerns have significantly boosted the development of FL, and especially PPFL.

Privacy Leakage

However, recent work has shown that FL is vulnerable to many data poisoning attacks impacting prediction accuracy, and inference attacks compromising data privacy [32, 54], raising concerns about using FL as a privacy-preserving mechanism alone.

Privacy-Preserving Mechanisms

We describe below the most popular privacy-preserving mechanisms applied to FL, including *Secure Multi-Party Computation* (SMC), *Homomorphic Encryption* (HE), and *Differential Privacy* (DP).

- **SMC-based PPFL.** SMC is an interactive cryptographic scheme that enables a set of parties to collaboratively compute a public function without revealing their private data [51]. It is mainly used in PPFL constructions to secure the aggregation protocol involving the local model updates [4]. While SMC offers strong privacy guarantees and high accuracy, it does not scale with the number of parties because of its significant computational overhead and communication costs.
- **HE-based PPFL.** HE schemes enable computations directly on encrypted data [39] and are widely used to protect data privacy by aggregating encrypted local model updates during the training process [38]. Although HE provides robust confidentiality guarantees and preserves utility, the computational cost induced by the encrypted operations significantly degrades performance.
- **DP-based PPFL.** DP is a rigorous mathematical framework for privacy-preserving data analysis that quantifies personal information disclosure in a dataset [15]. The main idea is to add sufficient noise to the training process to prevent data leakage from intermediate and aggregated model updates. DP has become a popular privacy-preserving approach to FL due to its natural trade-off between data privacy and utility which is controlled by the added amount of noise. We distinguish two applications of DP to FL:
    1. *Global Differential Privacy* (GDP) assumes a central server adding noise to the aggregated global model preventing malicious parties from inferring private information from the shared global model

[23]. However, GDP is vulnerable to a malicious server as it receives clear model updates from participants.

2. *Local Differential Privacy* (LDP) involves adding noise during local training to protect local model updates sent to the server and thus removes trust in the central aggregator [47, 10], as required in GDP. Nevertheless, LDP requires a larger total amount of noise than GDP and is therefore less accurate than GDP for an equivalent level of security, because LDP protects the privacy of each party's data separately. Intuitively, the local data in LDP is generally more granular and less informative than the aggregated data in GDP and is therefore more vulnerable to inference attacks.

1.2 Related Work

Since privacy-preserving cryptographic techniques, such as SMC and HE, have computational and communication overheads, and DP tends to degrade data utility, many recent hybrid PPFL constructions have been proposed. By combining these techniques, researchers hope to achieve better performance, privacy, and utility trade-offs than each method alone. Hybrid constructions typically include different combinations of SMC, HE, and DP.

Among them, Truex et al. [46] proposed a PPFL system combining SMC and LDP to train various ML models. Their solution calibrates the amount of local differential noise by a trust parameter that considers the number of honest, non-colluding parties. However, using the Paillier additive HE scheme [36] in their SMC protocol makes their system vulnerable to quantum attacks.

In addition, Hao et al. [25] designed a PPFL construction for large-scale AI by combining an LWE-based additive HE scheme with LDP, where local differential noise also depends on the ratio of colluding parties. Nevertheless, their experiments showed that the achieved accuracy of the NN seriously degrades when more than half of the participants collude.

Sav et al. [41] introduced POSEIDON, a novel PPFL system based on a combination of SMC and HE, better known as *Multi-Party Homomorphic Encryption* (MHE). This work supports privacy-preserving NN training in the passive-adversarial model allowing collusions of up to $K - 1$ parties by performing the whole training process under encryption. Although POSEIDON achieves high accuracy for a simple NN (3 layers), the significant computation and communication overhead caused by MHE makes it unusable for training more complex NNs.

1.3 Contributions

Our work makes the following contributions to the field of PPFL:

1. A novel hybrid approach to PPFL combining MHE and GDP that protects the confidentiality of the training data and intermediate model updates in a semi-honest threat model allowing collusions up to $K-1$ parties.
2. An analysis of the sensitivity and total privacy loss of our differentially private SGD algorithm implementing GDP.
3. A performance analysis of a fully encrypted FL for a 3-layer NN training using benchmarks from Lattigo [18], an open-source lattice-based MHE library in Go.
4. An experimental evaluation and benchmark of our hybrid solution on the MNIST and CIFAR-10 datasets, showing superior performance to fully encrypted approaches (POSEIDON) and comparable accuracy to non-private FL.
5. An investigation of the trade-offs between performance, privacy, and accuracy in applying DP to FL.

1.4 Outline

Chapter 2. Background

This chapter introduces theoretical background on deep learning and PETs for the rest of the thesis. We present SGD and NNs that we will use to evaluate our PPFL solution and describe the MHE scheme that our construction relies on.

Chapter 3. Differential Privacy

We first motivate the need for DP in the context of privacy-preserving data analysis, and explain its intuition. Next, we provide the mathematical framework that formally defines DP, introducing key definitions and describing mechanisms and techniques for achieving DP. We conclude this chapter by discussing the application of DP to ML.

Chapter 4. Privacy-Preserving Federated Learning

In this chapter, we first define the system and threat model we consider in the context of CTI sharing to design our PPFL solution, and evaluate FL as an individual privacy-preserving mechanism highlighting its vulnerabilities to various inference attacks. We then present a fully encrypted construction similar to POSEIDON that we will use as a comparison. Next, we describe

our PPFL solution combining MHE and DP, including our differentially private learning algorithm implementing GDP. We finally derive a bound on our algorithm's sensitivity and provide a lower bound for the differential noise variance.

Chapter 5. Experimental Evaluation

This chapter describes our evaluation setup and presents the experimental results of the two conducted experiments; we evaluate our hybrid construction by training a simple 3-layer NN and a more complex one on the MNIST and CIFAR-10 datasets distributed among 3 parties.

Chapter 6. Conclusion

We summarize our work and discuss the limitations of the practical application of DP and propose future work to improve the different trade-offs of our hybrid PPFL construction.

Chapter 2

Background

We first provide the necessary background on SGD and NNs and then present the MHE scheme on which our hybrid construction relies to perform privacy-preserving NN training in a federated K-party setting.

2.1 Deep Learning

Deep Learning (DL) is a subfield of ML that has recently gained popularity due to its state-of-the-art performance in solving complex problems in various fields, including computer vision or natural language processing, and enabling significant advances in image classification, speech and facial recognition, and language translation. Unlike traditional ML models that rely on manual feature engineering, deep learning algorithms train artificial NNs to automatically extract and learn meaningful features and intricate patterns from raw data through multiple layers, effectively representing high-dimensional data such as images, speech, and text. As a result, deep learning has become a popular tool for various applications in industry and academia.

2.1.1 Gradient Descent

Gradient Descent (GD) is a first-order optimization algorithm used as a ML training technique for finding the values of the model parameters that minimize a given loss function. It consists of iteratively adjusting the parameters toward the direction of the negative gradient of the loss function until we reach a minimum. We formally define gradient descent below.

Definition 2.1 (Gradient Descent Update) *Given a dataset* $D = \{x_1, ..., x_n\} \in \mathbb{R}^{n \times m}$*, a batch of training samples* $B \subseteq D$*, a loss function* $\mathcal{L}(\theta) = \frac{1}{|B|} \sum_{x \in B} \mathcal{L}(\theta, x)$ *with respect to the model parameters* $\theta \in \mathbb{R}^d$*, the gradient descent function* f_{θ_t} *updates the parameters* θ *of iteration* $t \in [T]$ *as follows:*

$$f_{\theta_t} : \mathbb{R}^d \to \mathbb{R}^d$$
$$\theta_t \mapsto \underbrace{\theta_t - \frac{\eta}{|B|} \sum_{x \in B} \nabla_\theta \mathcal{L}(\theta, x)}_{\theta_{t+1}}$$

where $\eta \in \mathbb{R}_+$ *is the learning rate which determines the step size of the update, and* $\nabla_\theta \mathcal{L}(\theta, x)$ *is the gradient of the loss function* $\mathcal{L}(\theta)$*.*

Specifically, the loss function $\mathcal{L}(\theta)$ provides a measure of how well the model fits the dataset and represents the penalty for mismatching the training data. The goal of learning is to find the model parameters θ that produce an acceptable loss.

2.1.2 Neural Networks

Neural Networks (NNs) are DL models that enable the extraction of relatively complex non-linear relations from the input data. They have many applications, such as data analysis, pattern recognition, and forecasting. NNs consist of several layers of neurons performing linear and non-linear transformations iteratively on their input data. A weight that adjusts during the learning process is assigned to each edge connecting neurons of different layers. Each training iteration comprises a *feed-forward* step (forward pass) updating the weights and a *backpropagation* step (backward pass) updating the gradients.

Multi-Layer Perceptrons (MLPs) are well-known *Fully-Connected* (FC) NNs composed of an input layer, one or multiple hidden layer(s), and an output layer, where each neuron in one layer is connected to all neurons in the next layer. We represent FC layer operations as matrix multiplications.

Formally, considering an iteration t, we denote by a matrix W_j^t the weights between two layers j and $j+1$, and by a matrix A_j^t the activation of the neurons in layer j. Each training iteration consists of the feed-forward and backpropagation phases, which repeat until the convergence of a set of weights that produce accurate predictions about the training data.

Forward Pass

The forward pass divides into two steps: a linear combination of the weights with the activation values of the previous layer

$$U_j^t = W_j^t \cdot A_{j-1}^t$$

and an evaluation of the activation function ϕ to compute the values of the neurons for each layer

$$A_j^t = \phi(U_j^t).$$

Backward Pass

Backpropagation is a method that applies gradient descent to minimize the prediction error of each iteration by adjusting the model weights accordingly. We provide below the update function of mini-batch GD in matrix form, where a random batch of sample inputs B is used in each iteration.

$$W_{j+1}^t = W_j^t - \eta \nabla W_j^t$$

where η is the learning rate and ∇W_j^t denotes the gradient of the loss function (error matrix E) with respect to the model weights and computed as

$$\nabla W_j^t = \frac{\partial E}{\partial W_j^t}.$$

We note that backpropagation requires several matrix and transpose operations applied to vectors and matrices.

2.2 Privacy-Enhancing Technologies

The need for *Privacy-Enhancing Technologies* (PETs) stems from the growing concern for personal privacy in the digital age. With the rapid growth of cyber technologies, large amounts of sensitive information are being collected, stored, and processed by various organizations and entities. As a result, there is a growing risk of data breaches, identity theft, and privacy violations.

By implementing PETs, organizations and individuals can gain more control over their personal information and enjoy greater privacy in their online activities. They ensure secure and responsible management of personal information and can thus help restore trust in the digital world.

2.2.1 Secure Multi-Party Computation

First introduced in 1986 by Yao [52], *Secure Multi-Party Computation* (SMC) is a well-studied and highly relevant problem in cryptography that we can state as follows.

Definition 2.2 (Secure Multi-Party Computation Problem) *A set of K participants $\{P_1, ..., P_K\}$, each owning private data $x_1, ..., x_K$ respectively, wants to jointly compute the value of a public function over their private inputs $F(x_1, ..., x_K)$, while keeping their own inputs private.*

Solutions to this problem are of great interest for securing distributed data-sharing scenarios where the participants do not fully trust each other but must collaborate on a computation. SMC protocols, enabling collaborative computation among multiple parties, have many real-world applications, including secure voting, financial services, healthcare, and in our case, *Privacy-Preserving Machine Learning* (PPML). Generally, SMC protocols are interactive and satisfy the next two properties.

1. **Input privacy.** No party should be able to infer more information about private data held by the other parties (from messages sent during protocol execution) than what is deducible from the output of the joint computation.
2. **Correctness.** The result of the joint computation is correct and consistent with the agreed public function and private inputs of the parties.

Adversarial Model

The security of SMC protocols is defined in terms of the capabilities and behavior of the modeled adversary and assumptions made about the parties. In the multi-party setting, the adversary may participate in or control the internal parties of the protocol. In the context of CTI sharing, we consider the *semi-honest model with dishonest majority* that assumes the following:

- **Private Inputs.** Each party has a private input that it wishes to keep confidential.
- **Computational Resources.** Each party has access to sufficient computational resources to perform the computations required by the protocol.
- **Communication Channels.** Parties can communicate with each other over a secure channel that is not susceptible to eavesdropping, interception, or tampering by the adversary.
- **Majority Control.** The adversary controls a majority of the parties. This assumption is realistic in real-world scenarios, as an attacker may be able to compromise a large number of devices or nodes in a network.
- **Semi-Honest Behavior.** Dishonest parties strictly follow the protocol as specified but may attempt to infer sensitive information about the private inputs of other parties from the received messages during protocol execution. Semi-honest behavior is also known as passive, rational, or honest-but-curious.
- **Colluding Parties.** Collusion is defined as any cooperative behavior between several parties controlled by the adversary to break the security guarantees of the protocol. Corrupted parties may collude by

sharing their private inputs and intermediate results to derive sensitive information about the private data of honest parties.

- **Limited Computation.** The adversary is computationally bounded, i.e., it can only perform polynomial-time computations and is, therefore, unable to break cryptographic schemes in a reasonable time.

Implementations

The predominant implementation of generic SMC solutions uses *secret sharing* schemes that generate shares of the private inputs and randomly distribute them among the parties. Each share is a piece of information that contains a partial representation of a party's private data but leaks no information about the private input itself. Then, the parties jointly perform the computation on the shares to get the final result without revealing anything about their private data. Intuitively, security is achieved because any set of shares looks randomly distributed. However, current approaches have high communication overhead because they generally require per-party communications that increase at least linearly with the number of parties. Therefore, this quadratic factor quickly becomes a bottleneck as we scale the number of parties.

2.2.2 Fully Homomorphic Encryption

Homomorphic Encryption (HE) enables computations on encrypted data, unlocking a wide range of privacy-preserving applications in many fields, such as healthcare, insurance, and cyber defense. The most powerful form of homomorphic encryption is known as *Fully Homomorphic Encryption* (FHE), which allows the evaluation of arbitrary circuits composed of multiple types of gates of unbounded depth.

Hardness Assumption

The security of modern FHE schemes relies on the hardness of the *Decision Ring-Learning-With-Errors* (Decision-RLWE) problem which has brought HE from being practical to being efficient.

Definition 2.3 (RLWE Distribution) *Let $\mathcal{R}_q = \mathbb{Z}_q[X]/f(X)$ be a quotient ring with modulus $q \geq 2$ and cyclotomic polynomial f of degree N, and χ be an error distribution over $\mathcal{R}_q$. Considering a secret $s \leftarrow\!\!\$\ \mathcal{R}_q$, an RLWE distribution $A^q_{s,\chi}$ samples arbitrarily many independent samples $(a, b = a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q$ where $a \leftarrow\!\!\$\ \mathcal{R}_q$ and $e \leftarrow\!\!\$\ \chi(\mathcal{R}_q)$.*

Definition 2.4 (Search-RLWE Problem) *For a random but fixed secret $s \leftarrow\!\!\$\ \mathcal{R}_q$, and given arbitrary many independent samples $(a, b = a \cdot s + e) \leftarrow A^q_{s,\chi}$, recover s with non-negligible probability.*

Definition 2.5 (Decision-RLWE Problem) *Given arbitrary many independent samples* $(a, b) \in \mathcal{R}_q \times \mathcal{R}_q$*, distinguish between the RLWE distribution* $A^q_{s,\chi}$ *and the uniform distribution over* $\mathcal{R}_q \times \mathcal{R}_q$ *with non-negligible probability.*

Leveled Homomorphic Encryption

A weaker form of homomorphic encryption is called *Leveled Homomorphic Encryption* (LHE) and supports the evaluation of arbitrary circuits of predetermined multiplicative depth. Specifically, in leveled mode, each ciphertext is assigned a level corresponding to the number of homomorphic multiplications that can be performed on it. This way, for a ciphertext at an initial level L, an L-depth circuit can be evaluated at most.

Technical Challenges

- **Noise Growth.** As ciphertexts are inherently noisy, performing multiple homomorphic operations on them combines error terms of the two operands, which increases the noise magnitude in the resulting ciphertext. When the noise of a ciphertext exceeds a threshold determined by the scheme's parameterization, decryption correctness breaks, and no further operation can be performed on it, making it unusable. Noise growth depends on the specific HE scheme and homomorphic operations performed.

  Boostrapping reduces the amount of noise accumulated in a ciphertext, enabling further homomorphic operations to be performed on it. It refreshes a ciphertext by returning its associated level to its initial value, overcoming the computational limitations posed by noise growth. This technique, therefore, enables FHE schemes to support the homomorphic evaluation of arbitrary-length circuits. However, bootstrapping is a computationally intensive procedure that remains the major bottleneck in FHE implementations.

- **Ciphertext Growth.** Another limitation of HE schemes is the exponential growth in the size of the resulting ciphertext after sequential homomorphic multiplications. As the number of homomorphic multiplications increases, the size of the ciphertext can quickly become impractical, requiring additional storage and computing resources.

  Relinearization is a homomorphic primitive that limits the growth of ciphertext size after multiplication. It is used to keep the size of ciphertexts manageable to enable further computations on them.

FHE Schemes

Although Gentry's first FHE scheme [22] was far from practical, his major theoretical breakthrough paved the way for a long line of work. Since then, the research community has expended tremendous efforts to improve performance and make FHE schemes more efficient.

Modern and popular FHE schemes rely on the Decision-RLWE hardness assumption and include BFV [5, 20], BGV [6] and CKKS [9].

BFV & BGV. The Brakerski/Fan-Vercauteren (BFV) and Brakerski-Gentry-Vaikuntantan (BGV) schemes are part of the so-called second generation of FHE schemes. These cryptosystems typically operate on integers in the leveled mode and enable efficient packing of plaintext values in a single ciphertext for SIMD-like computations.

CKKS. In 2016, Cheon et al. proposed the Cheon-Kim-Kim-Song (CKKS) scheme, which supports floating-point arithmetic and operates on real and complex numbers with high precision while minimizing ciphertext size. In practice, CKKS, which belongs to the fourth generation, is more efficient than BFV and BGV due to its efficient bootstrapping procedure. It also benefits from efficient SIMD batch computations over real-number vectors.

2.2.3 Multi-Party Homomorphic Encryption

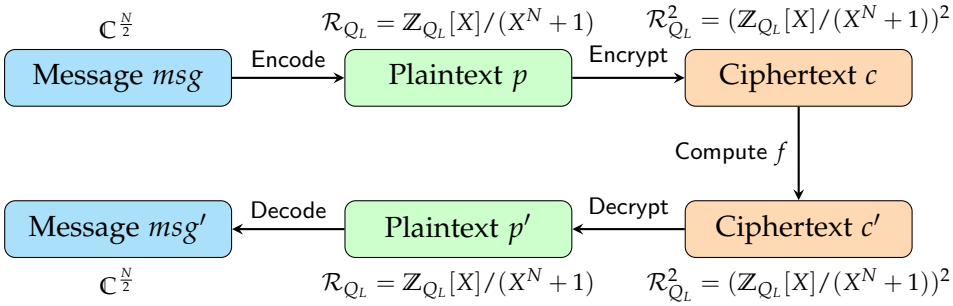
Our hybrid PPFL solution relies on the CKKS [9] variant of the MHE scheme proposed by Mouchet et al. [33]. We motivate the choice of this scheme in our construction as follows.

1. CKKS is particularly well suited for PPML involving real-valued data as it supports floating-point arithmetic.
2. CKKS relies on the hardness assumption of Decision-RLWE (Section 2.5.), making our system robust against quantum attacks.
3. MHE enables secure and flexible collaborative computations between parties while preserving the confidentiality of local data.

In this multi-party scheme, a collective public key is known by all parties, and the corresponding secret key is distributed such that decryption requires the collaboration of all parties.

Cryptographic Description. The plaintext and ciphertext spaces are defined by a cyclotomic polynomial ring $\mathcal{R}_{Q_L} = \mathbb{Z}_{Q_L}[X]/(X^N + 1)$, where N is a power-of-two integer, with $Q_L = \prod_{i=1}^{L} q_i$ where each moduli q_i is a unique prime, and Q_L is the ciphertext modulus at an initial level L. We denote by $\mathbf{c} = (c_0, c_1) \in \mathcal{R}_{Q_L}^2$ and $p \in \mathcal{R}_{Q_L}$ a ciphertext and a plaintext respectively.

For a ciphertext $\mathbf{c}$, we denote by L_c, S_c, L and S, the current level of $\mathbf{c}$, the current scale (bit precision) of $\mathbf{c}$, the initial level and the initial scale of a fresh ciphertext respectively, and we use similar notations for plaintexts.



We describe below the main functionalities that our system provides to enable PPFL. Operations expressed as $\prod$ are distributed K-party protocol executed among all the secret-key-holders, whereas the remaining operations can be performed locally by any party holding the collective public key.

- $\mathsf{SecKeyGen}(1^\lambda)$: Returns a set of secret keys $\{sk_i\}_{i=1}^K$
- $\prod_{\mathsf{PubKeyGen}}(\{sk_i\})$: Returns a collective public key pk
- $\mathsf{Encode}(msg \in \mathbb{C}^{\frac{N}{2}})$: Returns $\bar{p} \in \mathcal{R}_{Q_L}$ with scale S
- $\mathsf{Decode}(\bar{p} \in \mathcal{R}_{Q_{L_p}})$: Returns $msg \in \mathbb{C}^{\frac{N}{2}}$
- $\mathsf{Encrypt}(pk, \bar{p})$: Returns $\mathbf{c}_{pk} \in \mathcal{R}^2_{Q_L}$ with scale S
- $\prod_{\mathsf{Decrypt}}(\{sk_i\}, \mathbf{c}_{pk})$: Returns $\bar{p} \in \mathcal{R}_{Q_{L_\mathbf{c}}}$ with scale $S_\mathbf{c}$
- $\mathsf{Add}(pk, \mathbf{c}_{pk}, \mathbf{c}'_{pk})$: Returns $(\mathbf{c}+\mathbf{c}')_{pk}$ at $min(L_\mathbf{c}, L_{\mathbf{c}'})$, $max(S_\mathbf{c}, S_{\mathbf{c}'})$
- $\mathsf{Sub}(pk, \mathbf{c}_{pk}, \mathbf{c}'_{pk})$: Returns $(\mathbf{c}-\mathbf{c}')_{pk}$ at $min(L_\mathbf{c}, L_{\mathbf{c}'})$, $max(S_\mathbf{c}, S_{\mathbf{c}'})$
- $\mathsf{Mul}_{pt}(pk, \bar{p}, \mathbf{c}_{pk})$: Returns $(\bar{p} \cdot \mathbf{c})_{pk}$ at $min(L_p, L_\mathbf{c})$ and $(S_p \cdot S_\mathbf{c})$
- $\mathsf{Mul}_{ct}(pk, \mathbf{c}_{pk}, \mathbf{c}'_{pk})$: Returns $(\mathbf{c} \cdot \mathbf{c}')_{pk}$ at $min(L_\mathbf{c}, L_{\mathbf{c}'})$ and $(S_\mathbf{c} \cdot S_{\mathbf{c}'})$
- $\mathsf{Rescale}(\mathbf{c}_{pk})$: Returns $\mathbf{c}_{pk}$ at level $L_\mathbf{c} - 1$ with scale $\frac{S_\mathbf{c}}{q_{L_\mathbf{c}}}$
- $\mathsf{Relin}(\mathbf{c}_{pk} \in \mathcal{R}^3_{Q_{L_\mathbf{c}}})$: Returns $\mathbf{c}_{pk} \in \mathcal{R}^2_{Q_{L_\mathbf{c}}}$
- $\prod_{\mathsf{KeySwitch}}(\{sk_i\}, \mathbf{c}_{pk}, pk')$: Returns $\mathbf{c}_{pk'}$
- $\prod_{\mathsf{Bootstrap}}(\{sk_i\}, \mathbf{c}_{pk}, L_\mathbf{c}, S_\mathbf{c})$: Returns $\mathbf{c}_{pk}$ with initial level L and scale S

To control scale growth due to homomorphic multiplication, the $\mathsf{Rescale}(\cdot)$ primitive is applied to the resulting ciphertext after each multiplication. $\mathsf{Encode}(\cdot)$ enables packing several plaintext values into one ciphertext and processing them in parallel, benefiting from SIMD instructions.

Semantic Security

The CKKS-based MHE scheme is *IND-CPA secure*, if the advantage of any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ when playing the IND-CPA security game

Game IND-CPA$_{\mathcal{A}}(\lambda)$

$b \leftarrow\!\!\$\ \{0,1\}$
$\{sk_i\}_{i=1}^{K} \leftarrow\!\!\$\ \mathsf{SecKeyGen}(1^{\lambda})$
$pk \leftarrow \prod_{\mathsf{PubKeyGen}}(\{sk_i\})$
$(m_0, m_1) \leftarrow \mathcal{A}(pk)$
$\mathbf{c}_{pk} \leftarrow \mathsf{Encrypt}(pk, \mathsf{Encode}(m_b))$
$b' \leftarrow \mathcal{A}(pk, \mathbf{c}_{pk})$
return $b = b'$

is negligible, namely

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(\lambda) = \left| \Pr[\text{IND-CPA}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

Chapter 3

Differential Privacy

This chapter first introduces the context of privacy-preserving data analysis and motivates the need for DP. It then provides the mathematical framework, stating key definitions and presenting mechanisms and techniques for achieving DP. We conclude this chapter by discussing the application of DP to ML.

3.1 Motivation

3.1.1 Privacy-Preserving Data Analysis

Companies' growing use of customer data to improve their products and services has become a significant concern for privacy advocates in today's digital landscape. The collected data becomes increasingly personalized and sensitive, and if it were to leak, it could cause harm to individuals. On the other hand, customers are attracted by the high utility of these applications. They, therefore, are willing to share more data with the companies, which in turn become more powerful and valuable to them. Thus, there is tension between the needs of companies seeking to improve their products and services using customer data and the privacy concerns of those same customers.

Despite the best efforts of security researchers, more than the industry's standard best practices, such as data encryption, data retention, and access control, are needed to fully protect customer data's privacy. These practices are effective at protecting the data in transit or at rest but do not address the issue of what information ML models themselves can reveal about the underlying training data. These models can still expose sensitive information, even if the data is encrypted. The challenge, then, is to find ways to analyze data while preserving privacy.

Privacy-preserving data analysis has many applications across various industries, such as healthcare, financial services, marketing, social networking, and e-commerce. As the use of data continues to increase, the importance of protecting customer data is growing, making privacy-preserving data analysis a crucial aspect in different areas.

3.1.2 Data Anonymization

Data anonymization is a technique used to protect the privacy of individuals by removing or obscuring personally identifiable information (PII) from a dataset. This process aims to make data sharable and usable for research or analytics purposes while ensuring that individuals cannot be identified through the data. However, despite its advantages, data anonymization fails in practice because of the following attacks on privacy:

1. *Re-identification* occurs when a third party can match an anonymized dataset with another dataset that contains PII, such as the zip code, birthday, and gender, or with publicly available information to identify individuals in the anonymized dataset.
2. *Linkage* refers to linking records in a dataset to other records, even if they do not contain any direct PII. For instance, by combining information from multiple sources, an attacker could link anonymized data to an individual's identity.
3. *Inferences* are deductions made about an individual based on patterns or correlations in the data. For instance, an attacker could infer sensitive information about an individual based on location, demographics, or activity patterns.

The limitations of data anonymization have been demonstrated in various real-world cases, including:

- Netflix Prize: In 2006, Netflix held a competition to improve the accuracy of its movie recommendations. However, researchers were able to re-identify many individuals in the anonymized dataset using information from IMDb and other sources.
- AOL Search Data: In 2006, AOL released a dataset of search queries for 658,000 users, which was anonymized by removing usernames and other identifying information. Nevertheless, researchers were able to re-identify many users based on patterns in their search queries.
- Cambridge Analytica and Facebook: In 2014, Cambridge Analytica obtained data from millions of Facebook users through an app that promised to provide personality predictions. Although the data was anonymized, Cambridge Analytica could use it to profile and target users for political advertising.

While being a valuable tool to protect privacy, data anonymization does not provide any strong guarantee. Real-world cases demonstrate the limitations of data anonymization, highlighting the need for innovative privacy-preserving protections that apply to ML models.

3.1.3 Intuition

Differential privacy is a mathematical framework that provides a rigorous definition of privacy for data analysis. Its textual definition states that the outcome of an analysis is equally likely to occur, regardless of the participation of an individual in the dataset, and allows the knowledge of the general trends of the population without revealing private information about individuals. By plausible deniability, there is no privacy compromise if the information learned cannot be tied to a single individual. The goal is thus to protect the privacy of individuals by making it impossible or infeasible for an attacker to determine any individual's contribution to the results of data analysis.

However, privacy can still be compromised if a particular outcome becomes much more likely or unlikely after a data record has been removed or added to the dataset, indicating that information about a specific individual has leaked. Instead, the analyst should learn about the population as a whole and not about a specific individual in the dataset. DP thus guarantees that the information learned remains the same even if an individual is replaced by another random member of the population.

The key idea behind DP is to add a controlled amount of random noise to the data so that the privacy of individuals is protected while still allowing for meaningful analysis to be performed. In other words, DP ensures that the presence or absence of a single individual in the dataset does not significantly change the outcome of the analysis. This way, even if the data is disclosed, the privacy of individuals in the dataset is protected.

3.2 Mathematical Formalization

This section provides the mathematical framework of differential privacy.

3.2.1 Database

In data analysis, a database or dataset (used interchangeably) can be defined mathematically as a collection of tuples, where each tuple, also called *record*, is a structured list of attributes that usually describes the characteristics of an individual.

Definition 3.1 (Generic Database) *A generic database $D \in \mathcal{D}$ containing n records is defined as:*

$$D = \{x_1, ..., x_n\},$$

where x_i is an m-dimensional row vector representing the attributes of a single record, and where $\mathcal{D}$ is the space of all such databases.

We also define the notions of database size, and distance between databases, which will serve as building blocks for the following.

Definition 3.2 (Database Size) *The l_1-norm of a database D is denoted $\|D\|_1$. It measures the size of D, i.e. the number of records it contains, and is defined as:*

$$\|D\|_1 = \sum_{i}^{|D|} |x_i|.$$

Definition 3.3 (Distance Between Databases) *The distance between two databases $D, D' \in \mathcal{D}$, denoted $\|D - D'\|_1$, is a measure of how many records differ between both databases and is defined as:*

$$\left\|D - D'\right\|_1 = \sum_{i}^{\max |D|, |D'|} |x_i - x'_i|.$$

To formally define DP, we will mainly be interested in *adjacent* databases, namely a pair of databases that only differ in a single record.

Definition 3.4 (Adjacent Databases) *Two databases $D, D' \in \mathcal{D}$ are called adjacent if their l_1-distance is bounded by 1, that is:*

$$\left\|D - D'\right\|_1 = \sum_{i}^{\max |D|, |D'|} |x_i - x'_i| \leq 1.$$

3.2.2 Statistical Query

A statistical query to a database refers to a mathematical operation performed on the database that extracts statistics about the underlying population represented by the data.

Definition 3.5 (Statistical Query) *Given a database $D \in \mathcal{D}$, a statistical query can be defined as a function $f \colon \mathcal{D} \to \mathbb{R}^d$, where $f(D)$ provides statistical information about the underlying data distribution of the database.*

An example of a statistical query could be a function computing the mean of a particular attribute, such as the average salary or age, of individuals in employment records.

3.2.3 Definitions

The first definition of DP has been stated in 2006 by Dwork et al [14]. Its formal definition involves a randomized algorithm that we define as follows.

Definition 3.6 (Randomized Algorithm) *For $a \in A$, a randomized algorithm $\mathcal{M}: A \to B$ outputs $\mathcal{M}(a) = b$ with probability $Pr[\mathcal{M}(a) = b]$.*

Randomized algorithms can be viewed as deterministic algorithms taking two inputs: a database and a random bit string. The definition of DP includes a probability operator over the randomness of the random bit string (internal randomness of the algorithm), holding a fixed database. The input space is the space of all possible databases, while the output space is the space of database query results. In deep learning, it is often a space of learnable model parameters (e.g. NN weights).

Definition 3.7 (ε-Differential Privacy) *A randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is ε-differentially private if for all adjacent databases $D, D' \in \mathcal{D}$, and for any subset of outputs $S \subseteq \mathcal{R}$:*

$$Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} Pr[\mathcal{M}(D') \in S],$$

where $\varepsilon > 0$ is the privacy loss of the randomized algorithm $\mathcal{M}$.

Here, the *privacy loss* ε intuitively quantifies information leakage. When ε vanishes, we have zero privacy loss (perfect privacy) because the two probabilities are equal. However, maximum privacy is achieved at the cost of adding so much noise that the result is no longer useful. It is thus possible to quantitatively compare different ε values for different randomized algorithms. Intuitively, the closer ε is to zero, the closer the two probability distributions are, and the higher the level of privacy.

The limitation of *pure ε-DP* is that events with tiny probability (which are negligible in real-world applications) can dominate the privacy analysis. This motivates us to move to a more relaxed notion of differential privacy, such as *approximate (ε, δ)-DP*, which is less sensitive to low-probability events.

Definition 3.8 ((ε, δ)-Differential Privacy) *A randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is (ε, δ)-differentially private if for all adjacent databases $D, D' \in \mathcal{D}$, and for any subset of outputs $S \subseteq \mathcal{R}$:*

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

where $\varepsilon > 0$ is the privacy loss of the randomized algorithm $\mathcal{M}$.

Here, δ represents an upper bound on the probability that a differentially private algorithm is allowed to fail. If $\delta = 0$, (ε, δ)-DP is equivalent to ε-DP. As mentioned above, *pure ε-DP* ($\delta = 0$) takes into account all events

with non-zero probability, even the very unlikely ones, meaning that events characterized by a small probability are given unduly large consideration, even though they may never happen in practice. Instead, *approximate* (ε, δ)-*DP* does not consider events with probability less than δ, which informally states that ε-DP holds with probability $1 - \delta$. In practice, we should keep δ negligible with respect to the inverse of the database size. A common heuristic to choose δ for a database with n records is $\delta \in O(\frac{1}{n})$.

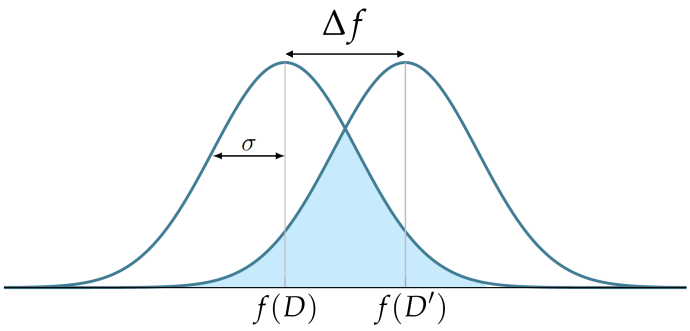
3.2.4 Sensitivity

The sensitivity of a query is a crucial concept in differential privacy that helps evaluate the worst-case privacy loss induced by the presence of individuals in the dataset. Specifically, the sensitivity measures the extent to which its result changes when a single data record is added or removed from the database. In other words, it captures the maximum amount of privacy loss that occurs when we replace an individual in the database. The intuition is that the more sensitive a query is, the more the query result changes, and the more privacy is compromised.

Its purpose is to determine the amount of noise that needs to be added to the query result to protect the privacy of individuals in the dataset, as the sensitivity quantifies the maximum impact of a data record on the outcome of a query. Therefore, by adding noise proportional to the sensitivity of the query, we ensure that the privacy of individuals in the database is protected.

Definition 3.9 (l_2-Sensitivity) *Given a statistical query $f : \mathcal{D} \to \mathbb{R}^d$ mapping a database to a d-dimensional real value, and adjacent databases $D, D' \in \mathcal{D}$, the l_2-sensitivity Δf is defined as:*

$$\Delta f = \max_{D,D'} \left\| f(D) - f(D') \right\|_2.$$



3.2.5 Mechanisms

Mechanisms are the basic building blocks to make advanced real-world applications differentially private.

Definition 3.10 (Gaussian Distribution) *A random variable X has a Gaussian distribution with mean μ and variance σ, if its probability density function $\mathcal{N}(\mu, \sigma^2)$ is:*

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{|x-\mu|}{\sigma}\right)^2}.$$

We define a *Gaussian mechanism* as follows.

Definition 3.11 (Gaussian Mechanism) *Given any function $f\colon \mathcal{D} \to \mathbb{R}^d$, the Gaussian mechanism is defined as*

$$\mathcal{M}_f(D, \varepsilon) = f(D) + (X_1, ..., X_d)$$

where X_i are independent and identically distributed random variables drawn from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$.

The Gaussian mechanism approximates an arbitrary function with a multi-dimensional output by adding noise sampled from the normal distribution where the noise variance is calibrated to the l_2-sensitivity of the function.

Theorem 3.12 (Gaussian Mechanism) *Given an arbitrary $\varepsilon \in (0, 1)$, and any function $f\colon \mathcal{D} \to \mathbb{R}^d$, a Gaussian mechanism $\mathcal{M}_f$ with parameter $\sigma \geq \frac{\Delta f \sqrt{2 \log \frac{1.25}{\delta}}}{\varepsilon}$ is (ε, δ)-differentially private.*

Proof See Theorem A.1 [15]. □

3.2.6 Composability

A crucial property of differentially private algorithms is their behavior under *composition*, which is independent of their actual implementation. Composition states that by running multiple differentially private algorithms on the same database, the resulting composed algorithm is also differentially private, albeit with some degradation in the privacy parameters (ε, δ). Specifically, the composition property is quantitative, meaning that the differential privacy guarantee of the composed algorithm depends on the number of differentially private algorithms and their privacy parameters. The exact relationship between these quantities can be relatively complex, and various composition theorems provide bounds on the global privacy parameters based on the parameters of the differentially private subroutines.

This property is critical to the success of DP, particularly for algorithm design, as it allows modular constructions of complex differentially private

algorithms from basic differentially private subroutines and helps analyze the privacy guarantee of such composed algorithms.

We will consider *sequential adaptive queries* performed on the same database for the next two reasons. First, this setting accounts for a realistic adversary that adapts its sequential queries to the same database based on all the previous output statistics she has received. Second, sequential adaptive queries fit well the iterative nature of gradient descent training, as each iteration is a function of the model parameters of the previous iteration.

Definition 3.13 (Sequential Adaptive Composition) *Let $D \in \mathcal{D}$ be a database, and $\mathcal{M}_i \colon \mathcal{D} \times \prod_{j=0}^{i-1} \mathbb{R} \to \mathbb{R}$ be an adaptive mechanism, taking the output results $x_0, ..., x_{i-1}$ of previous mechanisms as input. The composition of k adaptive mechanisms $\mathcal{M}_i(D, x_0, ..., x_{i-1}) = x_i$, denoted by $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$, is called a k-fold sequential adaptive composition.*

Basic Composition

Many composition theorems already exist in the literature. Of these, the simplest is known as *Basic Composition* and informally states that the privacy parameters of sequential adaptive mechanisms add up under composition.

Theorem 3.14 (Basic Composition [13]) *Let $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$ be a k-fold sequential adaptive composition of (ε, δ)-differentially private mechanisms. Then, $\mathcal{M}$ is a most $(k\varepsilon, k\delta)$-differentially private.*

Proof See Appendix A.1. □

Basic Composition shows a worst-case linear degradation of the global privacy parameters with the number of adaptive mechanisms in the composition. Improving this bound is of great interest because we could get more utility from any differentially private algorithm under the same privacy guarantees. We present below a direct consequence of Theorem 3.14 that enables a direct derivation of the privacy loss ε of an individual mechanism $\mathcal{M}_i$ depending on the *global* privacy loss ε_g of a k-fold sequential adaptive composition $\mathcal{M}$.

Corollary 3.15 (Basic Composition [13]) *Suppose a $(\varepsilon_g, \delta_g)$-differentially private k-fold sequential adaptive composition $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$. Then, each mechanism $\mathcal{M}_i$ is $(\frac{\varepsilon_g}{k}, \frac{\delta_g}{k})$-differentially private.*

Advanced Composition

Dwork, Rothblum, and Vadhan [16] showed that if we are willing to tolerate a tiny increase in the δ term, the privacy loss only degrades proportionally to $\sqrt{k}$ instead of k (Theorem 3.14), where k is the number of adaptive mechanisms composed sequentially. This result was later improved by Kairouz,

Oh, and Viswanath [27] and then clearly stated in the lecture notes of Adam Smith and Johnathan Ullman [42].

Theorem 3.16 (Advanced Composition [27, 42]) *For all $\varepsilon, \delta, \delta' \geq 0$, the class of (ε, δ)-differentially private mechanisms satisfies $(\varepsilon', k\delta + \delta')$-differential privacy under k-fold adaptive composition for:*

$$\varepsilon' = \varepsilon\sqrt{2k\log\frac{1}{\delta'}} + k\varepsilon\frac{e^{\varepsilon} - 1}{e^{\varepsilon} + 1}.$$

Proof See Section 3. [42]. □

Theorem 3.16 improves Basic Composition by showing that privacy degrades by a function of $O(\sqrt{k\log\frac{1}{\delta'}})$, which is an improvement if $\delta' = 2^{-O(k)}$. A direct consequence of Theorem 3.16 expresses the privacy loss ε of an individual mechanism $\mathcal{M}_i$ depending on the global privacy loss ε_g of a k-fold sequential adaptive composition $\mathcal{M}$.

Corollary 3.17 (Advanced Composition [16]) *Considering $\varepsilon_g, \varepsilon \in (0, 1)$ and $\delta, \delta' > 0$, the k-fold sequential adaptive composition $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$ satisfies $(\varepsilon_g, k\delta + \delta')$-differential privacy, if each mechanism $\mathcal{M}_i$ is (ε, δ)-differentially private, where*

$$\varepsilon = \frac{\varepsilon_g}{2\sqrt{2R\log\frac{1}{\delta'}}}$$

and

$$\delta = \frac{\delta_g - \delta'}{k}.$$

Proof See Appendix A.2. □

3.2.7 Privacy Amplification by Subsampling

In this section, we now focus on mechanisms using data subsampling and evaluate how it impacts privacy. The intuition behind *privacy amplification by subsampling* is that running a differentially private algorithm on a random subset of the data introduces additional uncertainty regarding the inclusion or exclusion of individuals in the analysis, which significantly boosts privacy. In particular, this uncertainty makes it more challenging for potential attackers to carry out generic attacks, ultimately contributing to privacy protection.

Lemma 3.18 (Privacy Amplification by Subsampling) *Suppose $\varepsilon \in (0, 1)$, a database D, a sample $S \subseteq D$ including each record of D with probability $q \in (0, 1)$, and an (ε, δ)-differentially private mechanism $\mathcal{M}$, then $\mathcal{M}(S)$ is $(2q\varepsilon, q\delta)$-differentially private.*

Proof See Appendix A.3. □

While subsampling is mainly used in statistical analysis, it also finds its applications in differentially private deep learning, particularly in SGD, where we select a random subset of the training data, also known as mini-batch, rather than using the entire dataset for training. Subsampling thus reduces the computational cost of learning and improves the privacy guarantee of the training algorithm.

3.3 Differentially Private Learning

In recent years, ML applications have become widespread in various fields, leading to a surge in research on the security and privacy of these methods. Researchers have identified several critical vulnerabilities and attacks associated with ML models, prompting discussions about developing appropriate defenses. Among the attacks that compromise the privacy of training data, model inversion [21] and membership inference [53] have received considerable attention.

As DP has emerged as a popular defense mechanism to mitigate these attacks, we present the different steps to make a ML task differentially private.

3.3.1 Data & Model Release

The research community addresses privacy issues in ML applications from two different angles.

1. **Data Release.** This approach sanitizes the original data by removing or hiding sensitive information, allowing for its publication and model training without privacy concerns. Privacy-preserving data release techniques include basic anonymization, which generally does not guarantee privacy as discussed in Section 3.1.2, to other advanced methods that transform data points or generate new synthetic data.

   While data release algorithms allow training ML models on released data without privacy constraints, implementing these methods on real-world datasets with complex data types is challenging and often results in weak privacy guarantees and poor predictive performance.

2. **Model Release.** This line of work provides a privacy guarantee for the model parameters before releasing it by enforcing privacy during training for each model update.

   Privacy-preserving model release methods offer less flexibility than data release techniques as they release a specific trained model. Nevertheless, model release is preferred because these algorithms enjoy

better performance and accuracy, are generally easier to implement, and provide stronger privacy guarantees than data release.

3.3.2 Differentially Private Gradient Descent

Most research on private ML focuses on privacy-preserving model release algorithms. In 2016, Abadi et al. [1] contributed significantly by proposing a novel *Differentially Private Stochastic Gradient Descent* (DP-SGD) algorithm to train deep learning models while protecting data privacy. In addition, they introduced the *Moments Accountant* (MA) technique which has since been extensively used in modern PPML to limit the cumulative privacy loss, providing tighter bounds than *Advanced Composition*. There are essentially two strategies to protect the privacy of the training data in gradient descent.

Output Perturbation adds differential noise to the final model parameters resulting from the training process, treating the whole procedure as a black box. However, since a rigorous characterization of the dependence of these final parameters on the training data is generally not available, the noise is therefore selected based on the worst-case analysis, which adds overly conservative noise to the final parameters, destroying the utility of the learned model.

Therefore, we prefer a more fine-grained and sophisticated technique, called *Gradient Perturbation*, in which we seek to control the influence of the original data during the training process, precisely in each gradient descent update. It considers each intermediate model update a sensitive release and distorts each computed gradient with differential noise. This approach is followed by Adabi et al. but also in previous works [43].

The DP-SGD algorithm is essential for understanding our contributions in the next chapter, and we briefly describe it in the pseudo-code below (adapted from Adabi et al.).

Algorithm 1 Differentially Private Gradient Descent

Inputs
Dataset $D = \{x_1, ..., x_N\}$, Loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$
Parameters
Dataset size n, Number of local iterations T, Batch size B, Learning rate η, Clipping constant C, Noise variance σ
Initialize θ_0 randomly
for $t \in [T]$ **do**
  Batch
  Randomly sample a batch B_t
  with probability $q = \frac{B}{N}$
  Gradient
  for $i \in B_t$ **do**
    $\boldsymbol{g}_t(x_i) \leftarrow \nabla_\theta \mathcal{L}(\theta_t, x_i)$
  Clipping
  $\bar{\boldsymbol{g}}_t(x_i) \leftarrow \boldsymbol{g}_t(x_i) \cdot \max\left(1, \frac{C}{\|\boldsymbol{g}_t(x_i)\|_2}\right)$
  Noise
  $\tilde{\boldsymbol{g}}_t \leftarrow \frac{1}{B}(\sum_i \bar{\boldsymbol{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \boldsymbol{I}))$
  Descent
  $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\boldsymbol{g}}_t$
Return θ_T

(ε, δ)-DP Guarantee

As every intermediate model θ_t is considered a sensitive release, we must first ensure that each step of Algorithm 1 is (ε, δ)-DP. To do so, we follow the next procedure.

1. Consider each training step as the following model update function (Definition 2.1)

$$\begin{aligned} f_{\theta_t} : \mathbb{R}^d \times \mathcal{D} &\to \mathbb{R}^d \\ (\theta_t, D) &\mapsto \underbrace{\theta_t - \frac{\eta}{B}\sum_{x \in B} \overline{\nabla}_\theta \mathcal{L}(\theta, x)}_{\theta_{t+1}} \end{aligned}$$

   where $\overline{\nabla}_\theta$ denotes the clipped gradient with l_2-norm and constant C.

2. Upper-bound the l_2-sensitivity of f_{θ_t} (Definition 3.9), as follows

$$\Delta f_{\theta_t} = \max_{D, D'} \left\| f_{\theta_t}(D) - f_{\theta_t}(D') \right\|_2.$$

3. Apply a Gaussian mechanism to each training step f_{θ_t} (Definition 3.11), as below
$$\mathcal{M}_{f_{\theta_t}}(D, \varepsilon) = f_{\theta_t}(D) + \mathcal{N}(0, \sigma \mathbb{I}^d)$$
with noise variance $\sigma \geq \frac{\Delta f \sqrt{2 \log \frac{1.25}{\delta}}}{\varepsilon}$ (Theorem 3.12). This guarantees that each step is (ε, δ)-differentially private with $\varepsilon \in (0, 1)$ and negligible δ.

4. If the batch is randomly sampled in f_{θ_t} (with probability q), use *privacy amplification by subsampling* (Lemma 3.18) to achieve $(2q\varepsilon, q\delta)$-DP for each step.

5. Use the composition property to bound the total privacy loss over all iterations.
    - Basic Composition (Theorem 3.14)
    - Advanced Composition (Theorem 3.16)

Chapter 4

Privacy-Preserving Federated Learning

We first define the system and threat model we consider in the design of our PPFL solution and evaluate FL as an individual privacy-preserving mechanism highlighting its vulnerabilities to various inference attacks. Then, we present a fully encrypted construction similar to POSEIDON [41] that we use as a comparison. Next, we describe our PPFL solution combining MHE and GDP, including our differentially private learning algorithm. We finally derive the sensitivity of our learning algorithm and provide a lower bound for the differential noise variance.

4.1 Threat Model

In this section, we build on the adversarial model presented in Section 2.2.1 to define the threat model in the CTI sharing scenario, including the stakeholders' roles and security assumptions. A stakeholder may endorse multiple overlapping roles within the infrastructure of a given organization.

1. **Data Providers.** This includes any organization involved in CTI sharing scenarios, such as government intelligence agencies or cyber defense groups, providing authentic and reliable cyber threat data.
2. **Data Processors.** They perform data processing and analysis either internally or externally, relying on a private or public cloud.
3. **Data Consumers.** Data analysts or information security experts can use the final trained model to predict or detect future threats within these organizations.

We assume all stakeholders to be *semi-honest* (passively adversarial) as in Section 2.2.1 and recall that we allow for $K-1$ collusions between any participating organizations. We recall that parties controlled by the adversary follow the protocol but can share their inputs and intermediate results dur-

ing the training phase to extract information about the private data of non-colluding parties.

4.2 System Model

We consider a federated framework in which K parties collectively train a NN model on their local private data and describe the associated system model.

- The system is a fully-distributed and topology-agnostic network with K parties, e.g., a fully-connected network, or a star topology in which each party communicates with a central server.
- Parties involved in the federated training process want to preserve the confidentiality of their local data, the intermediate model updates, and the final model.
- When encryption is used, each party has a collective public and private key pair to encrypt intermediate model updates generated during the learning process.
- Once the training process is complete, a querier – either one of the K parties or an external entity – can request and use the final trained model to get prediction results on its private data.

4.3 Centralized Learning

In the centralized approach, depicted in Figure 4.1, all participants involved in the CTI sharing scenario send their private data to a central server responsible for training the model on the aggregated data and broadcasting the final model to all parties.

While centralized learning allows parties to outsource training computation to a powerful cloud server and prevent data sharing between them, this architecture suffers from the following drawbacks:

- **Communication overhead.** Data transfer from participants to the central server can be slow for large datasets.
- **Computational scalability.** As the central server must process large amounts of data, centralized learning may not scale to the number of parties or large datasets.
- **Trust requirement.** Parties must trust the central server and completely lose control over their private data, which leads to legal or ethical concerns about data ownership.

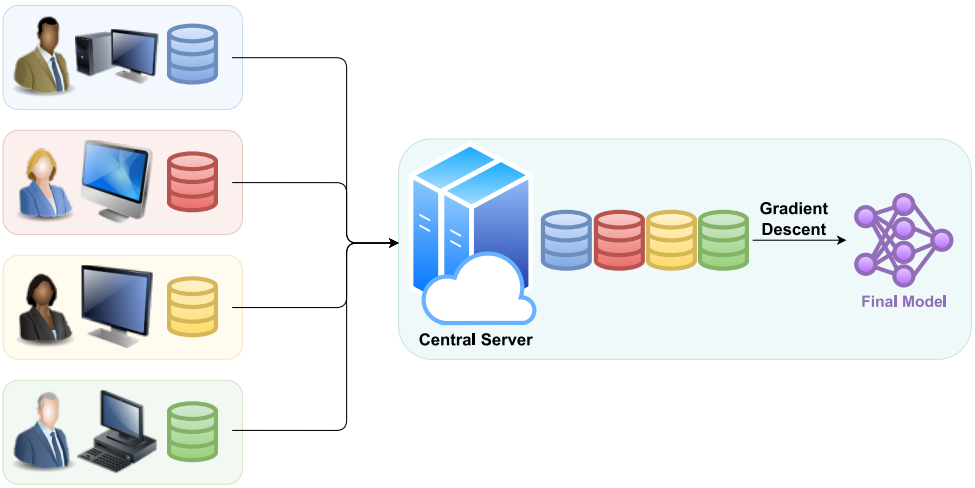


Figure 4.1: Centralized Learning.

- **Single point of failure.** As the private data of all parties is stored in one location, if the central server fails or is compromised, the entire system becomes unusable, and all data can also be compromised.

These limitations justify the need for the popular federated architecture.

4.4 Vanilla Federated Learning

First proposed by McMahan et al. [29] in 2016, FL is a decentralized approach to ML that enables multiple parties to train models collaboratively without exchanging or centrally storing their data. The main principle is that parties only send their local model updates to the central server rather than their entire dataset. Thus, training is distributed among the parties and performed locally instead of centralizing data in a training server.

In gradient descent learning, parties initially receive a pre-trained or random model from the server and locally perform GD training on this model using their private data for several local iterations. Then, each party sends its updated model to the central server, aggregating all local models into a global one and distributing it to the parties, as represented in Figure 4.2. This process is repeated over several communication rounds until the final global model converges.

FL presents several advantages over the centralized approach. First, training locally and sending models instead of datasets reduces data transfer and communication rounds, enhancing server-side performance and scalability. Second, as data is stored locally, it does not transfer or trivially disclose the raw data and it provides more data control to the parties.

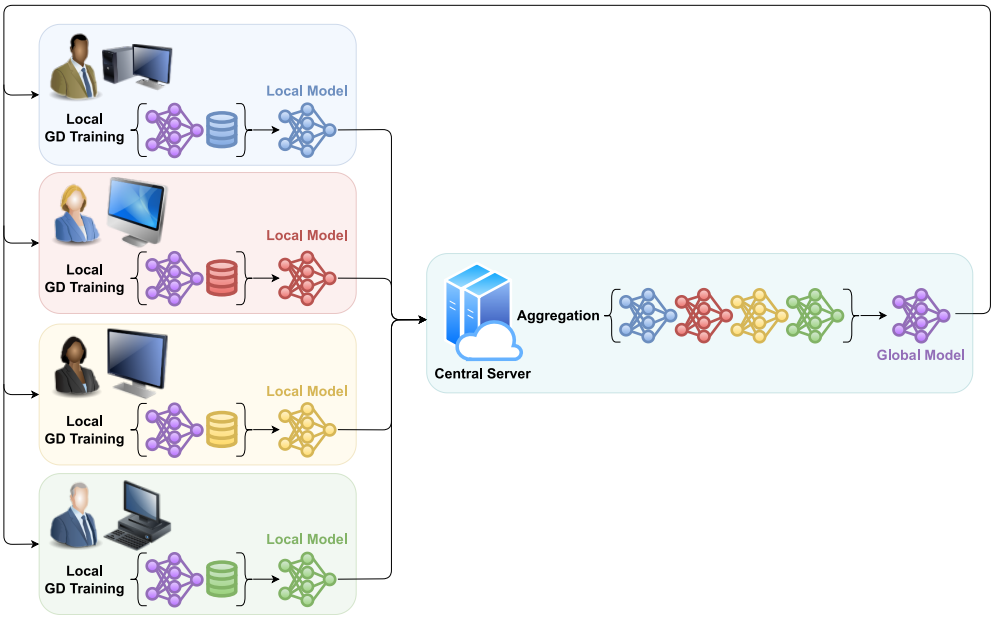


Figure 4.2: Vanilla Federated Learning.

4.4.1 Privacy Leakage

Recent work [35, 30, 49, 55, 40] has revealed that FL fails to preserve the privacy of the training data, as released model information and query results can lead to serious privacy leakage through various inference attacks.

Adversarial Setting

We distinguish two types of actors with different capabilities as potential adversaries.

1. **Insiders.** Internal adversaries include the participants and the central server responsible for orchestrating the entire training process.
2. **Outsiders.** External adversaries comprise model consumers accessing and querying the final model and eavesdroppers intercepting communication between the parties and the central server.

We consider internal actors and model consumers as potential adversaries, excluding eavesdroppers, as we assume secure communication channels between the parties and the server (using TLS 1.3). In addition, we suppose *passive white-box attacks* in which adversaries can observe the model information exposed during the training phase, including local and aggregated model weights (as shown in Figure 4.3), as well as the final model, but also query results on the final model, obtained in the inference phase.

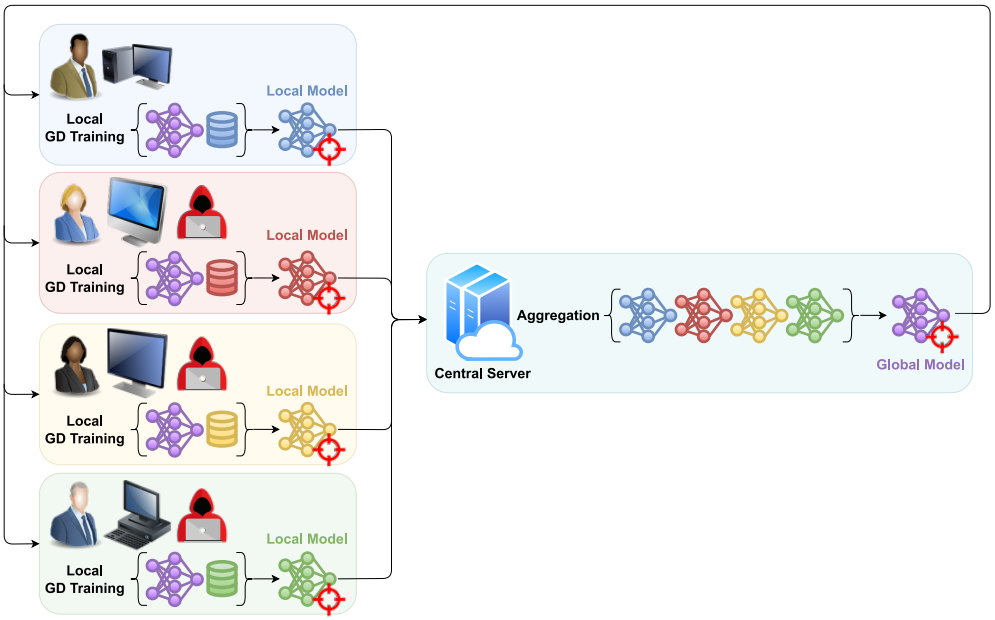


Figure 4.3: Compromised Federated Learning: Passive adversary controlling $K-1$ parties.

Passive adversaries can then use the model-related information within their reach to conduct inference attacks to extract and reveal sensitive information about the private training datasets of participants.

4.4.2 Inference Attacks

FL is vulnerable to different inference attacks that are classified as follows [17].

Membership Inference. This family of inference attacks aims to determine the presence of a data sample in the training dataset that generated a given model [35]. For example, these attacks can determine whether records of a specific patient were used to train a classifier related to a particular disease.

Properties Inference. These attacks infer auxiliary properties from the training dataset by guiding the global model to learn separable data representations with and without the target property [30]. Property inference attacks assume the existence of auxiliary training datasets labeled with the target property.

Class Representatives Inference. This type of attack attempts to extract synthesized generic data samples, called class representatives, instead of the actual data samples used in the training phase [49]. A famous example of this attack, also known as model inversion, was proposed by Fredrikson

et al. [21], who extracted sensitive features from input images and generated face images from facial recognition models with relatively high accuracy.

Data Sample & Label Inference. These are the most powerful attacks as they recover the original training samples and corresponding labels, notably from publicly shared model gradients [55], or loss function [40].

These attacks are particularly successful in iterative algorithms where the number of features is equal to or larger than the size of the dataset, such as deep learning, because a large amount of aggregated data (potentially leaking private data) is released at each iteration.

The current situation raises regulatory concerns about using FL as a standalone privacy-preserving technique, especially in light of the European Union's *General Data Protection Regulation* [48]. As a result, the research community has devoted considerable efforts to prevent such attacks by enforcing the privacy of the training and inference phases.

4.4.3 Privacy Objectives

Our main objective is to enable the privacy-preserving training and evaluation of NNs in the above system and threat model. In particular, we want to protect the parties' private data, as well as intermediate and final model weights vulnerable to the aforementioned inference attacks. Therefore, during the training and prediction phases, we aim to ensure the following:

- **Data Confidentiality.** *No party P_i should learn more information about the private input data X_j of any other party P_j (for $j \neq i$).*
- **Model Confidentiality.** *No party P_i should be able to infer information about the intermediate local model weights of other parties P_j (for $j \neq i$).*

4.5 Encrypted Federated Learning

This section presents a fully encrypted variant of FL that provides robust privacy-preserving guarantees by leveraging HE during the training and inference phases to prevent the inference attacks mentioned above. The construction depicted in Figure 4.4 is a simplified description of POSEIDON [41] that guarantees data and model privacy by using the CKKS-based MHE scheme introduced in Section 2.2.3.

Specifically, intermediate model updates stay encrypted under the parties' collective public key throughout the training process. The homomorphic computation property of CKKS enables operations required for NN training involving the parties' local data and the encrypted model weights. The

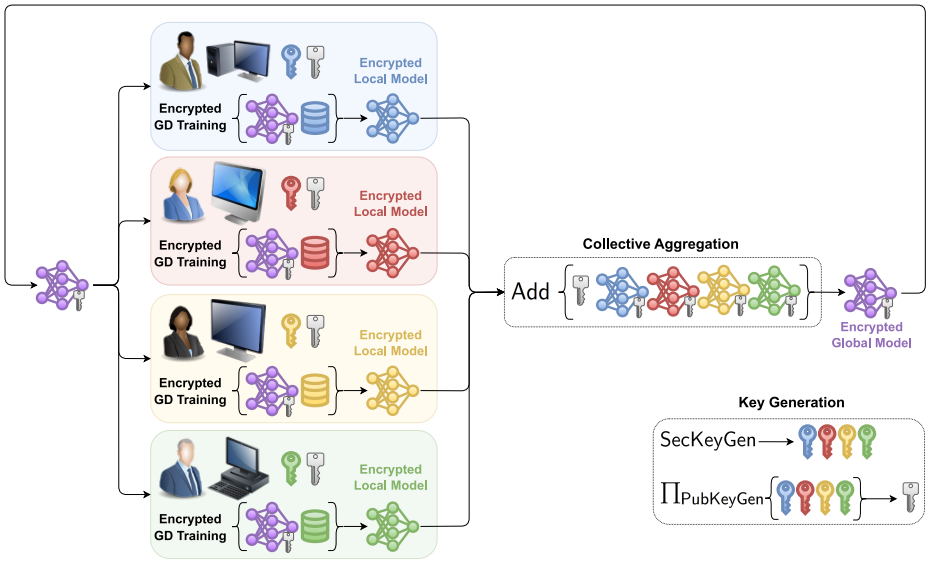


Figure 4.4: Fully Encrypted Federated Learning.

scheme's distributed key-switching functionality $\prod_{\mathsf{KeySwitch}}(\cdot)$ enables oblivious inference on the encrypted final model by collectively re-encrypting the prediction results with the querier's public key.

POSEIDON implements several optimizations to reduce the performance overhead induced by HE, such as efficient packing schemes to take advantage of SIMD instructions' parallelism and polynomial approximations of multiple activation functions and their derivatives to enable their evaluation under encryption.

However, POSEIDON requires computationally costly bootstrapping operations to refresh ciphertexts and be able to perform complex operations involved in the forward and backward passes of NN training. In particular, POSEIDON trains a 3-layer NN on the MNIST dataset ($m = 784$ features and $n = 60000$ samples) distributed among $K = 10$ parties in 1 hour and 28 minutes. While this execution time may be acceptable in ML, it can become impractical for training complex NNs with potentially hundreds of hidden layers.

4.6 Hybrid Federated Learning

This section presents our main contribution to the field of PPFL. We propose a hybrid construction, represented in Figure 4.5, combining the CKKS-based MHE scheme described in Section 2.2.3 and DP to achieve data and model confidentiality in the above system and threat model.

In our approach, DP guarantees the privacy of local training data when the

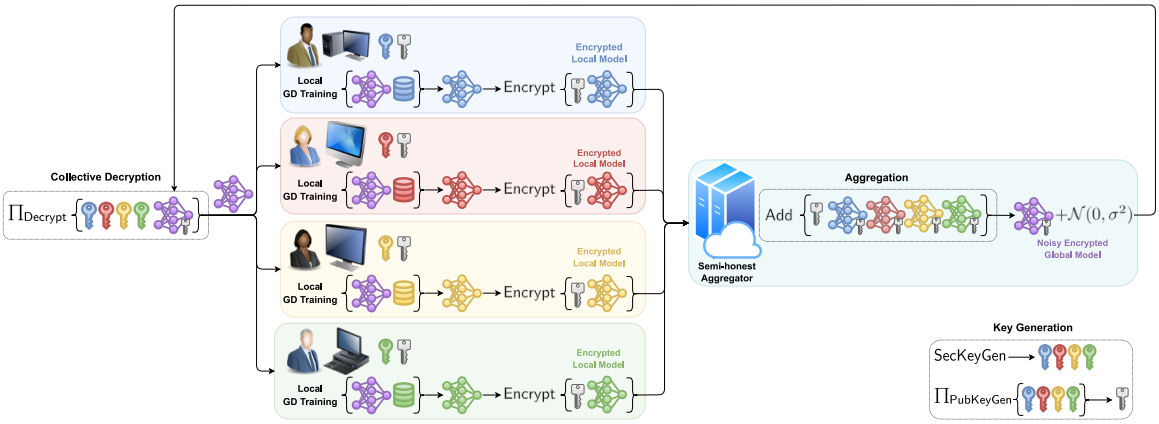


Figure 4.5: Hybrid Federated Learning.

intermediate model weights are released, while the homomorphic property of CKKS enables secure aggregation by the central server, ensuring confidentiality of the local contributions to the global model.

In the following sections, we first motivate our choices for the design of our solution, then provide a detailed description of our federated gradient descent algorithm, and finally explain how to achieve the DP guarantee introduced in Chapter 3.

4.6.1 Algorithm Design

When applying DP to FL, we recall the two options for adding differential noise in the learning process: LDP and GDP (introduced in Section 1.1.2). In the former, the noise is added to the local models by the parties, whereas it is added to the aggregated model by the central server in the latter. We chose GDP in the design of our construction for the following reasons.

1. **Model accuracy.** GDP requires adding less noise than LDP for an equivalent security level, thus resulting in higher prediction accuracy.
2. **Colluding Parties.** In LDP, corrupted parties can exchange the noise each one has added to its local model and subtract the total amount from the aggregated model to infer sensitive information. PPFL constructions implementing LDP [47, 25] consider a collusion parameter, accounting for the number of colluding parties, that increases the magnitude of local noise and thus degrades utility. GDP is robust to the collusion of $K-1$ parties, as the magnitude of differential noise that the central server adds to the aggregated model is unknown to them.

One advantage of LDP over GDP is the removal of the central aggregator from the architecture. However, the presence of a central server is not a

drawback per se, as vanilla FL assumes a central entity by default. Furthermore, to reduce the trust GDP typically requires from the central aggregator, the model aggregation is performed under encryption using HE. In this way, the central server cannot observe the parties' local consider updates, and we can therefore model it as passively adversarial.

4.6.2 Hybrid Federated Gradient Descent

We describe every step of our hybrid PPFL solution in Algorithm 2. We denote by $\theta^k_{r,t}$ the model weights of party P_k at round r and local iteration t, and by $\boldsymbol{\theta^k_{r,t}}$ its encrypted version.

1. Once the key generation procedure is complete, every party P_i owns a secret key sk_i and a collective public key pk.
2. Initially, either the central aggregator or each party individually initializes the model parameters θ_0 randomly.
3. At the beginning of each round, the parties collectively decrypt the global aggregated model of the previous round using the set of their secret keys $\{sk_i\}_{i=1}^K$.
4. Each party locally performs gradient descent on its private dataset involving the next steps:
   a) Randomly sample a disjoint batch B_t such that $B_t \cap \bigcup_{i=1}^{t-1} B_i = \emptyset$.
   b) Compute the gradient of the loss function for the batch B_t with respect to the model parameters θ.
   c) Clip the gradient vector $\boldsymbol{g}_t$ such that its l_2-norm is at most a threshold C.
   d) Update the model parameters $\theta_{r,t}$ by gradient descent for the next local iteration.
5. At the end of the local training, each party encrypts its local model weights θ_{r,T_k} and sends it to the central aggregator.
6. The central server receives the encrypted local model updates $\boldsymbol{\theta^k_{r,T_k}}$ of each party and produces an aggregated global model $\boldsymbol{\theta_r}$ using the homomorphic arithmetic operations of the CKKS-based MHE scheme.
7. Finally, the central aggregator adds Gaussian differential noise where the variance σ is calibrated at the sensitivity of a single round.

The following section details the derivation of the sensitivity for Algorithm 2, which provides an lower bound on the Gaussian noise variance required for the (ε, δ)-DP guarantee.

Algorithm 2 Federated Differentially Private Gradient Descent

Inputs
Dataset $D = \{x_1, ..., x_n\}$, Loss function $\mathcal{L}(\theta) = \frac{1}{n}\sum_i \mathcal{L}(\theta, x_i)$

Parameters
Dataset size n, Number of parties K, Number of rounds R,
Number of local iterations T, Batch size B, Learning rate η,
Clipping constant C, Noise variance σ

procedure KEY GENERATION(λ)
    $\{sk_i\}_{i=1}^K \leftarrow \mathsf{SecKeyGen}(1^\lambda)$
    $pk \leftarrow \prod_{\mathsf{PubKeyGen}}(\{sk_i\})$

for $k \in [K]$ **do**
    Initialize θ_0^k randomly
for $r \in [R]$ **do**
    for $k \in [K]$ **do**
        procedure CLIENT($\boldsymbol{\theta_{r-1}}$)
            $\theta_r \leftarrow \prod_{\mathsf{Decrypt}}(\{sk_i\}, \boldsymbol{\theta_{r-1}})$
            for $t \in [T_k]$ **do**
                Batch
                Randomly sample a disjoint batch B_t
                Gradient
                for $i \in B_t$ **do**
                    $g_t(x_i) \leftarrow \nabla_\theta \mathcal{L}(\theta_{r,t}^k, x_i)$
                Clipping
                $\bar{g}_t(x_i) \leftarrow g_t(x_i) \cdot \max\left(1, \frac{C}{\|g_t(x_i)\|_2}\right)$
                Descent
                $\theta_{r,t+1}^k \leftarrow \theta_{r,t}^k - \frac{\eta}{B}\sum_i \bar{g}_t(x_i)$
            Encryption
            $\boldsymbol{\theta}_{\boldsymbol{r},\boldsymbol{T_k}}^{\boldsymbol{k}} \leftarrow \mathsf{Encrypt}(pk, \theta_{r,T_k}^k)$
    procedure CENTRALAGGREGATOR($\{\boldsymbol{\theta}_{\boldsymbol{r},\boldsymbol{T_k}}^{\boldsymbol{i}}\}_{i=1}^K$)
        Encrypted Aggregation
        $\boldsymbol{\theta_r} \leftarrow \mathsf{Add}(pk, \{\boldsymbol{\theta}_{\boldsymbol{r},\boldsymbol{T_k}}^{\boldsymbol{i}}\}_{i=1}^K)$
        $\boldsymbol{\theta_r} \leftarrow \mathsf{Mul}_{pt}(pk, \frac{1}{K}, \boldsymbol{\theta_r})$
        Differential Noise
        $\boldsymbol{\theta_r} \leftarrow \boldsymbol{\theta_r} + \mathcal{N}(0, \sigma^2 C^2 \boldsymbol{I})$
Return $\boldsymbol{\theta_R}$

4.6.3 Differentially Private Guarantee

To provide Algorithm 2 with an (ε, δ)-differentially private guarantee protecting the intermediate and final model weights, we follow the same procedure as in Section 3.3.2. We aim to find the tightest lower bound on the variance of Gaussian noise that the central server adds to the aggregated model to minimize accuracy loss while preserving privacy. To do so, we apply a Gaussian mechanism to each round and model Algorithm 2 as a sequential adaptive composition for which we can derive an (ε, δ)-DP guarantee.

We begin by formally defining each round of our federated differentially private gradient descent algorithm as follows.

Definition 4.1 (Model Update Function) *Assuming K parties, every round r of Algorithm 2 can be represented as the model update function f_{θ_r} producing a global aggregated model by averaging the K local models and defined as:*

$$f_{\theta_r} : \mathbb{R}^d \times \mathcal{D}^K \to \mathbb{R}^d$$

$$(\theta_r, \{D_i\}_{i=1}^K) \mapsto \underbrace{\frac{1}{K} \sum_{k \in K} \underbrace{\left[\theta^k_{r,T_k-1} - \frac{\eta}{B} \sum_{x \in B} \overline{\nabla}_\theta \mathcal{L}(\theta^k_{r,T_k-1}, x) \right]}_{\theta^k_{r,T_k}}}_{\theta_{r+1}}$$

where D_i is the local dataset of party P_i, and $\overline{\nabla}_\theta$ denote the clipped gradient with l_2-norm and constant C.

Sensitivity

The next step is to derive an upper bound for the l_2-sensitivity (Definition 3.9) of the model update function defined above.

Lemma 4.2 (Global Sensitivity) *Given the model parameters θ_r, the datasets $\{D_i\}_{i=1}^{K-1} \in \mathcal{D}^{K-1}$ and $D, D' \in \mathcal{D}$ that differ in a single record $x_D \neq x_{D'}$ with $\|D - D'\|_2 = 1$, the sensitivity Δ_f of the model update function $f_{\theta_r} : \mathbb{R}^d \times \mathcal{D}^K \to \mathbb{R}^d$ is defined as:*

$$
\begin{aligned}
\Delta_f &= \max_{D,D'} \left\| f_{\theta_r}(\theta_r, \{D_i\}_{i=1}^{K-1} \cup D) - f_{\theta_r}(\theta_r, \{D_i\}_{i=1}^{K-1} \cup D') \right\|_2 \\
&= \max_{\theta,x} \left\| \frac{1}{K} \sum_{k \in K} \left[\theta^k_{r,T_k-1} - \frac{\eta}{B} \sum_{x \in B} \overline{\nabla}_\theta \mathcal{L}(\theta^k_{r,T_k-1}, x) \right] - \frac{1}{K} \sum_{k \in K} \left[\theta^k_{r,T_k-1} - \frac{\eta}{B} \sum_{x \in B'} \overline{\nabla}_\theta \mathcal{L}(\theta^k_{r,T_k-1}, x) \right] \right\|_2 \\
&= \max_{\theta,x} \frac{1}{K} \left\| \sum_{k \in K} \left[\theta^k_{r,T_k-1} - \frac{\eta}{B} \sum_{x \in B} \overline{\nabla}_\theta \mathcal{L}(\theta^k_{r,T_k-1}, x) \right] - \sum_{k \in K} \left[\theta^k_{r,T_k-1} - \frac{\eta}{B} \sum_{x \in B'} \overline{\nabla}_\theta \mathcal{L}(\theta^k_{r,T_k-1}, x) \right] \right\|_2 \\
&= \max_{\theta,x} \frac{1}{K} \left\| \theta^k_{r,T_k-1} - \frac{\eta}{B} \sum_{x \in B} \overline{\nabla}_\theta \mathcal{L}(\theta^k_{r,T_k-1}, x) - \theta^k_{r,T_k-1} + \frac{\eta}{B} \sum_{x \in B'} \overline{\nabla}_\theta \mathcal{L}(\theta^k_{r,T_k-1}, x) \right\|_2 \\
&= \max_{\theta,x} \frac{1}{K} \left\| \frac{\eta}{B} \sum_{x \in B'} \overline{\nabla}_\theta \mathcal{L}(\theta_{r,T_k-1}, x) - \frac{\eta}{B} \sum_{x \in B} \overline{\nabla}_\theta \mathcal{L}(\theta_{r,T_k-1}, x) \right\|_2 \\
&= \max_{\theta,x} \frac{\eta}{BK} \left\| \sum_{x \in B'} \overline{\nabla}_\theta \mathcal{L}(\theta_{r,T_k-1}, x) - \sum_{x \in B} \overline{\nabla}_\theta \mathcal{L}(\theta_{r,T_k-1}, x) \right\|_2 \\
&= \max_{\theta,x_D,x_{D'}} \frac{\eta}{BK} \left\| \overline{\nabla}_\theta \mathcal{L}(\theta_{r,T_k-1}, x_{D'}) - \overline{\nabla}_\theta \mathcal{L}(\theta_{r,T_k-1}, x_D) \right\|_2 \\
&\leq \frac{2C\eta}{BK}
\end{aligned}
$$

Our derivation is consistent with previous work from Wei et al. [50].

Gaussian Mechanism

We then apply a Gaussian mechanism to each round of Algorithm 2, represented by f_{θ_r}, as below

$$\mathcal{M}_{f_{\theta_r}}(\theta_r, \{D_i\}_{i=1}^{K}, \varepsilon_r) = f_{\theta_r}(\theta_r, \{D_i\}_{i=1}^{K}) + \mathcal{N}(0, \sigma \mathbb{I}^d),$$

with noise variance $\sigma \geq \frac{\Delta f \sqrt{2 \log \frac{1.25}{\delta_r}}}{\varepsilon_r}$ (Theorem 3.12), $\varepsilon_r \in (0, 1)$ and negligible δ_r, ensuring that each round $\mathcal{M}_{f_{\theta_r}}$ is $(\varepsilon_r, \delta_r)$-differentially private.

Sequential Adaptive Composition

We model Algorithm 2 as a *sequential adaptive composition* (Definition 3.13) of Gaussian mechanisms $\mathcal{M} = (\mathcal{M}_R \circ ... \circ \mathcal{M}_1)$ and then use the composition property to derive the DP parameters $(\varepsilon_r, \delta_r)$ of each mechanism $\mathcal{M}_r$. In particular, given global target parameters $(\varepsilon_g, \delta_g)$ for the whole algorithm, we obtain two pairs of $(\varepsilon_r, \delta_r)$ parameters depending on whether we use

Basic Composition (Theorem 3.14) or *Advanced Composition* (Theorem 3.16), both of which providing a closed-form expression for the DP parameters of a single round.

$$(\varepsilon_r, \delta_r) \in \left\{ \underbrace{\left(\frac{\varepsilon_g}{R}, \frac{\delta_g}{R}\right)}_{\substack{\text{Basic Composition} \\ \text{Corollary 3.15}}}, \underbrace{\left(\frac{\varepsilon_g}{2\sqrt{2R\log\frac{1}{\delta'}}}, \frac{\delta_g}{R+1}\right)}_{\substack{\text{Advanced Composition} \\ \text{Corollary 3.17}}} \right\}$$

Gaussian Noise Variance

Given the sensitivity of a single round Δ_f computed above, and fixed global DP parameters $(\varepsilon_g, \delta_g)$, we derive a lower bound on the Gaussian noise variance using Theorem 3.12. Since our goal is to reduce the negative effect of noise on prediction accuracy, we choose the minimum noise variance that provides the DP guarantee to our hybrid federated gradient descent algorithm.

$$\begin{aligned}
\sigma &\geq \frac{\Delta_f \sqrt{2\log\frac{1.25}{\delta_r}}}{\varepsilon_r} \\
&\geq \min_{\varepsilon_g, \delta_g, R} \left\{ \underbrace{\frac{2RC\eta\sqrt{2\log\frac{1.25R}{\delta_g}}}{BK\varepsilon_g}}_{\text{Basic Composition}}, \underbrace{\frac{4C\eta\sqrt{2R\log\frac{(R+1)}{\delta_g}}\sqrt{2\log\frac{1.25(R+1)}{\delta_g}}}{BK\varepsilon_g}}_{\text{Advanced Composition}} \right\}
\end{aligned}$$

For a fixed sensitivity $\Delta_f = \frac{2C\eta}{BK}$, and fixed global DP parameters $(\varepsilon_g, \delta_g)$, the Gaussian noise variance σ only depends on the number of rounds R, as shown in Figure 4.6.

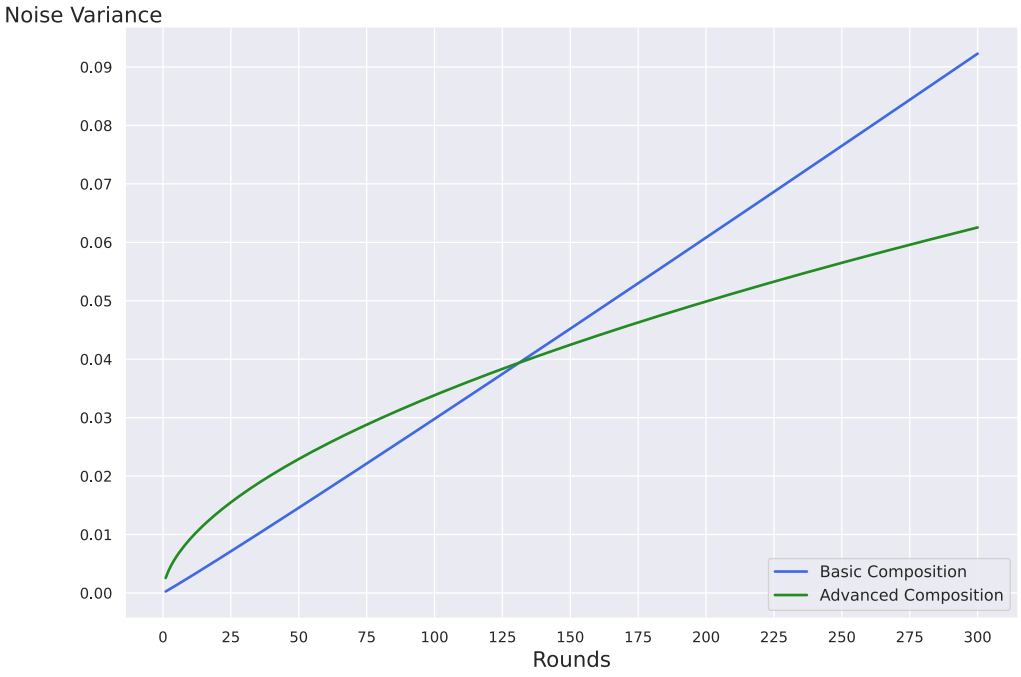


Figure 4.6: Noise variance with respect to the number of rounds R according to Basic and Advanced Composition ($\delta' = \delta_r$) for a global privacy loss $\varepsilon_g = 1$, negligible failure probability $\delta_g = 10^{-5}$, number of parties $K = 3$, batch size $|B| = 128$, clipping constant $C = 1$, and learning rate $\eta = 0.01$.

Chapter 5

Experimental Evaluation

This chapter describes our evaluation setup and presents the experimental results of our two experiments. As no CTI dataset met our requirements in terms of dataset size and class distribution, we decided to conduct a representative analysis on popular and publicly available datasets. Specifically, we evaluate our hybrid construction by training a simple 3-layer NN and a more complex one on the MNIST and CIFAR-10 datasets distributed among 3 parties.

5.1 Implementation

We implement our hybrid PPFL solution on top of PyTorch [37] and Lattigo [18], an open-source library for lattice-based cryptography in Go [12] using multi-party cryptographic primitives. Specifically, we build a decentralized system in which parties communicate over TCP via secure channels (TLS 1.3).

5.2 Experimental Setup

We leverage the computational capacity of Amazon Elastic Compute Cloud (Amazon EC2 [2]) to evaluate our construction in a virtual network (1 Gbps bandwidth) simulating a federated environment. This decentralized network consists of 3 CPUs, clocked at 3 GHz using 4 cores (Intel Xeon Scalable Broadwell E5-2686 v4) and 16 GB RAM, emulating $K = 3$ parties, and a single GPU (NVIDIA T4 Tensor Core) running at 2.5 GHz on 8 cores (Intel Xeon Scalable Cascade Lake P-8259L) and 16 GB RAM, that performs NN learning for all parties. One of the three nodes acts as the central aggregator, which receives the encrypted local models of the two remaining nodes and aggregates all local models under encryption.

Regarding the cryptographic parameters of the CKKS-based MHE scheme, we use 32-bit precision, a ring dimension $N = 2^{13}$, and number of levels $L = 6$.

5.3 Evaluation Metrics

We experimentally evaluate the performance, privacy, and accuracy of our hybrid PPFL solution according to the following metrics:

- **Performance** : Per-round execution time
- **Accuracy** : Prediction accuracy of the final model
- **Privacy** : Global (ε, δ) parameters

Baseline. We consider a non-private FL architecture similar to the one presented in Section 4.4. On the one hand, this approach does not implement any privacy-preserving mechanisms, such as HE or DP, and is therefore vulnerable to inference attacks. On the other hand, we can consider it as an ideal baseline precision and runtime as it does not suffer from any accuracy loss incurred by DP or computational overhead caused by HE.

Full Encryption. We compare our hybrid PPFL solution with the construction described in Section 4.5 (POSEIDON) to show the performance gain over fully encrypted approaches to FL. Since POSEIDON's implementation is no longer compatible with Lattigo's latest version, we provide an extrapolation of the execution time for the training phase of a 3-layer fully-connected NN on the MNIST dataset using the CKKS-based MHE scheme in Appendix A.4. We use benchmark runtimes for the different homomorphic matrix operations presented in Section 2.1.2, which consider Lattigo's latest optimizations. In addition, we assume that fully encrypted FL achieves similar accuracy as non-private FL constructions since HE has no impact on prediction accuracy.

5.4 Methodology

This section presents the procedure we follow for each experiment to evaluate our hybrid PPFL solution.

1. For the chosen NN model, we fix the optimal hyperparameters (learning rate η and batch size B) that achieve high accuracy for our baseline while minimizing the number of rounds R needed to reach convergence.
2. To obtain an optimal clipping constant C that achieves high accuracy while limiting the amount of differential noise in the DP setting, we

first set a relatively high clipping constant C so that the baseline accuracy is unaffected. Then, we gradually reduce the value of C for each run until prediction accuracy decreases slightly.

3. After fixing parameters η, B, and C that define sensitivity Δ_f, we apply DP to the NN model as described in Algorithm 2 by introducing global target DP parameters $(\varepsilon_g, \delta_g)$ in our setting. Typically, we set $\delta_g < \frac{1}{n}$ where n is the number of samples in the dataset, and training is said to be in *high privacy regime* when $\varepsilon_g \in (0, 1)$.

5.5 Empirical Results

This section reports on our experimental evaluation of our hybrid construction and results on two widely used and publicly available image datasets: MNIST and CIFAR-10.

From Algorithm 2, we recall that parties update the NN model using their whole local dataset, i.e., the model is trained on the entire dataset at each round, so one round corresponds to one epoch $(R = E)$.

In the following experiments, we normalize training image samples and distribute them equally and randomly among the participating parties with the same class distribution. Note that data and label distribution across parties and its effects on model accuracy are beyond the scope of this work.

Sensitivity. We emphasize the crucial role that sensitivity plays in the prediction accuracy of our hybrid approach. From Section 4.6.3, the sensitivity of Algorithm 2 is expressed as

$$\Delta_f \leq \frac{2C\eta}{BK}.$$

On the one hand, parameters η, B and C must be carefully tuned to obtain optimal performance and accuracy. On the other hand, they must also be chosen to minimize the sensitivity, because this quantity directly scales the amount of differential noise ($\sigma \propto \Delta_f$ in Section 4.6.3) and thus impacts prediction accuracy.

Rounds. Section 4.6.3 shows that the Gaussian noise variance is a function of the number of rounds. Specifically, σ is proportional to R in Basic Composition and $\sqrt{R}$ in Advanced Composition. Therefore, the number of rounds R must also be adjusted to allow a sufficient number of epochs to reach convergence while minimizing its effect on the amount of noise added at each round.

5.5.1 MNIST

In our first experiment, we train a simple 3-layer fully-connected NN with a single hidden layer of 92 neurons, using the sigmoid linear unit (SiLU) as activation function and cross-entropy loss on the MNIST dataset for handwritten digit recognition, which consists of $n = 60000$ gray-scale images of size 28x28 with $m = 28 \cdot 28 = 784$ features, classified into $h_l = 10$ classes.

Following the methodology of Section 5.4, we first fix $\eta = 0.01$ and $B = 128$ to reach a baseline accuracy of 96.2% in $R = 30$ epochs, and find $C = 0.5$ as an optimal clipping constant empirically. We set $\delta_g = 10^{-5}$ and consider different global privacy loss values in the high privacy regime $\varepsilon_g \leq 1$.

We present our experimental results for the MNIST experiment in Figure 5.1, which shows the evolution of the prediction accuracy as a function of the number of epochs (rounds), and summarize the obtained prediction accuracies in Table 5.1.

	Accuracy [%]
Baseline	96.2
Fully Encrypted	96.2
Hybrid ($\varepsilon_g = 1.0$)	93.4
Hybrid ($\varepsilon_g = 0.5$)	93.1
Hybrid ($\varepsilon_g = 0.1$)	90.6
Hybrid ($\varepsilon_g = 0.05$)	81.0
Hybrid ($\varepsilon_g = 0.01$)	50.8

Table 5.1: Prediction accuracies of the non-private (baseline), fully encrypted and hybrid constructions for different global privacy losses ε_g in the high privacy regime.

The above results demonstrate that our hybrid PPFL solution achieves similar accuracy to non-private FL in the high privacy regime ($\varepsilon_g = 1.0$) and relatively high accuracy ($> 90\%$) for $\varepsilon_g = 0.1$ which provides a solid confidentiality guarantee according to Definition 3.8.

We also show the increased performance of our hybrid construction over fully encrypted FL in Table 5.2. Regarding local training time, we consider a batch size $B = 128$ for the non-private (baseline) and hybrid constructions and $B = 256$ for the fully encrypted approach, as specified in Appendix A.4. The total training time for each FL construction is also provided. It includes the runtime of local training, encrypted aggregation, and collective decryption without considering the performance overhead generated by communicating the local model updates to the central server.

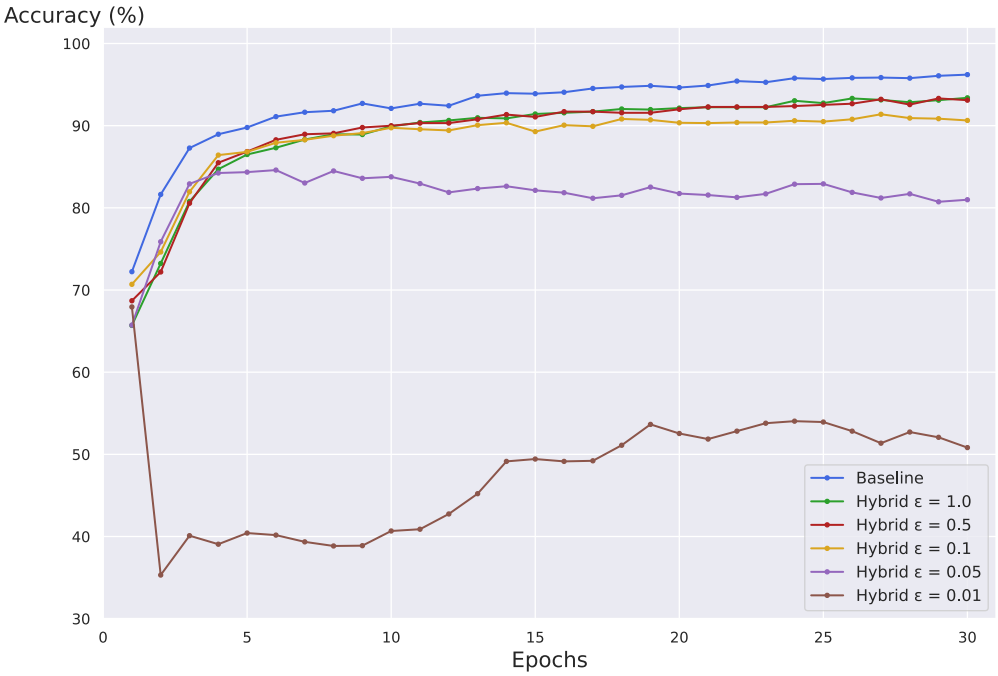


Figure 5.1: Evolution of the prediction accuracy of the non-private and hybrid constructions over the number of rounds for different global privacy losses ε_g.

	Execution Time $[sec]$			
	Local Training	*Encrypted Aggregation*	*Collective Decryption*	*Total (30 Rounds)*
Baseline	0.58	–	–	17.4 $[sec]$
Fully Encrypted	1159.4	2.01	0.11	9.67 $[hrs]$
Hybrid	0.61	1.89	0.12	78.6 $[sec]$

Table 5.2: Time measurements of the local training, encrypted aggregation of $K = 3$ intermediate local models, and collective decryption of the global aggregated model (including communication between parties).

Regarding local learning runtime, our hybrid algorithm achieves the same execution time as vanilla FL, and trains much faster than the fully encrypted construction. While the aggregation step is negligible in non-private learning, its encrypted version is the bottleneck in the hybrid solution. Since the NN model is relatively lightweight (654 KB), communications required in the collective decryption do not produce any overhead, and the whole operation is fast. The last column shows the total runtime of the train-

ing phase for $R = 30$ rounds, which includes the parallel local training of $\frac{N}{K} = \frac{60000}{3} = 20000$ image samples and the homomorphic aggregation and collective decryption operations.

5.5.2 CIFAR-10

Although the total training time of fully encrypted FL (9.67 hours) in the MNIST experiment may be acceptable for a ML task, it quickly explodes when considering deeper and larger NNs, making fully encrypted NN training impractical for more involved ML tasks without the help of hardware acceleration. In this second experiment, we demonstrate the practicality of our hybrid solution by training a complex NN on the CIFAR-10 dataset [28] composed of $n = 50000$ RGB image samples of $d = 3 \cdot 32 \cdot 32 = 3072$ features also categorized into $h_l = 10$ classes. More precisely, we train a CNN with an EfficientNetB2 architecture [44] with 473 layers, using the sigmoid linear unit (SiLU) as activation function and cross-entropy loss.

To improve our results, we introduce the following optimization concerning the sensitivity of our hybrid algorithm.

Layer-Specific Sensitivity. Looking at the weights of each layer, we notice that setting a global clipping constant C for all layers is sub-optimal because our algorithm often clips gradients of large-component layers (greater than C) and thus only considers gradients below this threshold. As a result, the model loses important information by clipping these large gradients, which results in a significant accuracy loss for the final model. To apply a more fine-grained clipping at the layer level, we introduce a *layer-specific sensitivity* by slightly modifying the derivation of Δ_f (Section 4.6.3). To do so, we pre-train our NN model θ on the public ImageNet dataset [11], and consider the norm of each layer $\ell_i \in \theta$. We modify the clipping value at line 23 of Algorithm 2 by $C \cdot \|\ell_i\|$, where the constant C acts as a scaling factor. The sensitivity of layer ℓ_i can then be expressed as

$$\Delta_f^i \leq \frac{2C\|\ell_i\|\eta}{BK}.$$

From Section 4.6.3, we hence obtain a *layer-specific noise variance* σ_i which defines the amount of differential noise to add to each layer's weights.

We follow the same procedure as before, i.e., we obtain 84.2% baseline accuracy in $R = 30$ epochs by fixing $\eta = 0.005$ and $B = 128$. As specified above, we find an optimal scaling factor $C = 0.01$ and set $\delta_g = 10^{-5}$.

Figure 5.2 shows our experimental results for the CIFAR-10 experiment using different global privacy loss values in the high privacy regime $\varepsilon_g \leq 1$. Prediction accuracies of the different scenarios are gathered in Table 5.3.

	Accuracy [%]
Baseline	84.2
Hybrid ($\varepsilon = 1.0$)	82.3
Hybrid ($\varepsilon = 0.5$)	82.7
Hybrid ($\varepsilon = 0.1$)	77.0
Hybrid ($\varepsilon = 0.05$)	13.1
Hybrid ($\varepsilon = 0.01$)	10.7

Table 5.3: Prediction accuracies of the non-private and hybrid constructions for different global privacy losses ε_g in the high privacy regime.

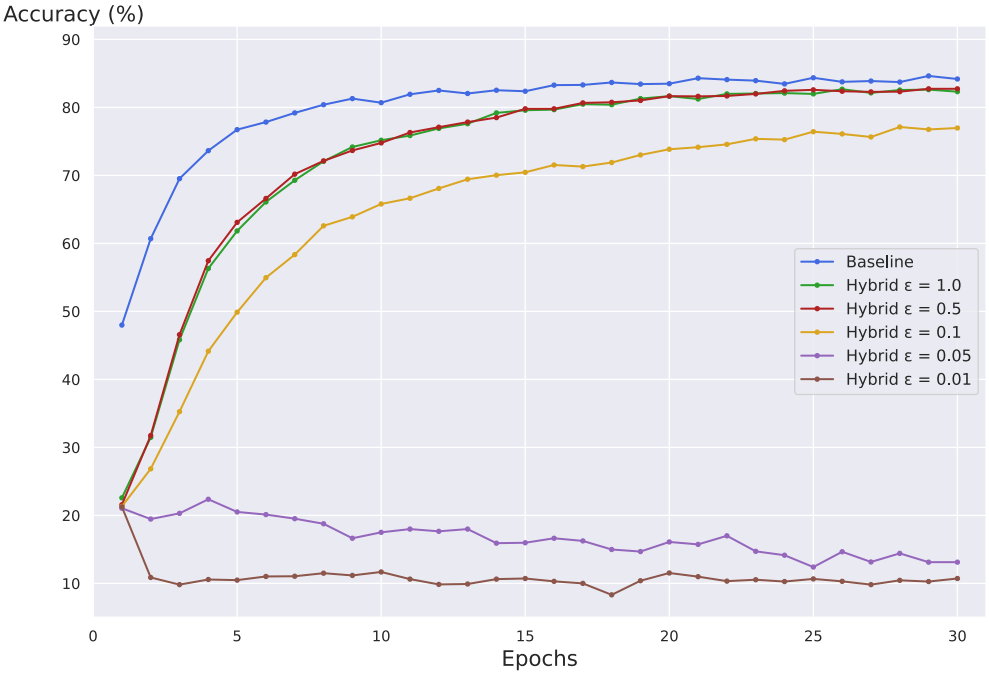


Figure 5.2: Evolution of the prediction accuracy of the non-private and hybrid constructions over the number of rounds for different global privacy losses ε_g.

Thanks to the optimization of the layer-specific sensitivity, our hybrid construction almost reaches the baseline accuracy for $\varepsilon_g = 0.5$ and still an acceptable accuracy (77.0%) for $\varepsilon_g = 0.1$. To the best of our knowledge, our hybrid construction is the first to obtain such results on the CIFAR-10 task while guaranteeing this level of privacy in the federated setting.

In a similar way, Table 5.2 summarizes and compares the performance of our hybrid algorithm against non-private FL.

	Execution Time [*sec*]			
	Local Training	*Encrypted Aggregation*	*Collective Decryption*	*Total* (*30 Rounds*)
Baseline	2.72	–	–	81.6
Hybrid	2.81	7.84	8.57	576.6

Table 5.4: Time measurements of the local training ($B = 128$), encrypted aggregation of $K = 3$ intermediate local models, and collective decryption of the global aggregated model (including communication between parties).

The model size (62.3 MB) is crucial in explaining these measurements. Indeed, the EfficientNetB2 model contains more than 7 millions weights that need to be encrypted and aggregated. This is because the complexity of the encrypted aggregation is linear with the model size, and collective decryption involves communication between parties which is also dependent on the model size. Nevertheless, the runtimes of these two homomorphic operations do not produce significant overhead, and the overall training time for 30 rounds remains practical.

Chapter 6

Conclusion

This final chapter summarizes our results, presents the limitations of our hybrid PPFL construction, and proposes future research directions to improve the trade-offs between performance, accuracy, and privacy.

6.1 Summary

Our main objective was to propose a PPFL solution to address the lack of confidentiality guarantee in CTI sharing and enable collective cyber defense between different actors of the cyber security landscape. To this end, we reviewed the literature on hybrid PPFL constructions based on different combinations of privacy-preserving mechanisms, such as SMC, HE and DP, and proposed a hybrid PPFL construction combining MHE and GDP to guarantee data and model confidentiality in the passive-adversary model.

Our contributions include a thorough analysis of the sensitivity of our hybrid algorithm as well as an evaluation of our PPFL solution on the popular MNIST and CIFAR-10 datasets. Our experimental results show that our hybrid construction achieves similar accuracy to non-private FL while ensuring a strong DP guarantee. Moreover, it enables privacy-preserving training of complex NNs, which remains impractical with current fully-encrypted FL implementations without any hardware acceleration.

6.2 Limitations

Differential privacy is of growing interest to the research community and has become a popular privacy-preserving standard for establishing formal privacy guarantees. The common way to improve the privacy-accuracy trade-off in iterative learning algorithms is to tighten the composition bounds of DP. In this sense, several variations of DP have been proposed, such as

(ε, δ)-*Differential Privacy* (introduced in Section 3.8), *zero-Concentrated Differential Privacy* (zCDP) [8] or *Rényi Differential Privacy* [31], which provide tight analysis of the cumulative privacy loss. These variants reduce the amount of noise that must be injected to satisfy given ε values, resulting in better utility.

Nevertheless, there is generally limited understanding of how these DP definitions truly impact utility in practice, and it remains unclear how much privacy leaks in adversarial scenarios. In addition, there is no concrete consensus or guidance on choosing appropriate privacy parameters (ε, δ). As a result, differentially private ML implementations tend to choose arbitrary large values for ε to achieve acceptable model utility, with little understanding of the impact of such choices on the meaning of DP guarantees. Furthermore, while increasing the privacy loss enhances model utility, it also increases the success rate of inference attacks.

In particular, Jayaraman and Evans [26] demonstrated that formal DP guarantees are way too conservative, in the sense that practical inference attacks are far from extracting the maximum leakage predicted by theory. Research is, therefore, needed to assess the limitations of inference attacks and provide a tighter estimations of the practical use of DP in privacy-preserving data analysis.

6.3 Future Work

While our proposal can successfully address the challenges of CTI sharing, there are several avenues for future study to expand on this work. We propose the following research directions that could enhance our hybrid construction in terms of performance, accuracy, and privacy.

1. A promising direction to improve the trade-off between privacy and accuracy is to investigate *Optimal Composition* [34], which provides a tighter bound than Basic and Advanced Composition on the privacy loss of a differentially private algorithm.

2. Another potential extension is to explore the impact on the different trade-offs of client subsampling by the central server. On the one hand, from Lemma 3.18, the privacy loss could be scaled down by the subsampling ratio, thus reducing the total amount of differentially-private noise. On the other hand, only a subset of the entire dataset is used for training at every round, which would require more rounds to reach convergence and thus increase noise variance.

3. Future research could also focus on implementing inference attacks against our PPFL construction to better understand the practical impact of different choices of ε for different variations of DP.

Appendix A

Appendix

A.1 Basic Composition

Theorem A.1 (Basic Composition [13]) *Let $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$ be a k-fold sequential adaptive composition of (ε, δ)-differentially private mechanisms. Then, $\mathcal{M}$ is a most $(k\varepsilon, k\delta)$-differentially private.*

Proof Let $D, D' \in \mathcal{D}$ be two adjacent databases, $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$ be a k-fold sequential adaptive composition of (ε, δ)-differentially private mechanisms, and $x \in \mathbb{R}^{k+1}$ be the output results of mechanims $\mathcal{M}_1, ..., \mathcal{M}_k$. Then,

$$
\begin{aligned}
\Pr[\mathcal{M}(D) = x_k] &= \prod_{i=1}^{k} \Pr[\mathcal{M}_i(D, x_0, ..., x_{i-1}) = x_i] \\
&\leq (e^{\varepsilon} \Pr[\mathcal{M}_1(D', x_0) = x_1] + \delta) \prod_{i=2}^{k} \Pr[\mathcal{M}_i(D, x_0, ..., x_{i-1}) = x_i] \\
&\leq e^{\varepsilon} \Pr[\mathcal{M}_1(D', x_0) = x_1] \prod_{i=2}^{k} (\Pr[\mathcal{M}_i(D, x_0, ..., x_{i-1}) = x_i]) + \delta \\
&\leq ... \\
&\leq e^{k\varepsilon} \prod_{i=1}^{k} \Pr[\mathcal{M}_i(D', x_0, ..., x_{i-1}) = x_i] + k\delta \\
&\leq e^{k\varepsilon} \Pr[\mathcal{M}(D') = x_k] + k\delta. \qquad \square
\end{aligned}
$$

A.2 Advanced Composition

Corollary A.2 (Advanced Composition [16]) *Considering $\varepsilon_g, \varepsilon \in (0,1)$ and $\delta, \delta' > 0$, the k-fold sequential adaptive composition $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$ satisfies $(\varepsilon_g, k\delta + \delta')$-differential privacy, if each mechanism $\mathcal{M}_i$ is (ε, δ)-differentially private, where*

$$\varepsilon = \frac{\varepsilon_g}{2\sqrt{2R \log \frac{1}{\delta'}}}$$

and

$$\delta = \frac{\delta_g - \delta'}{k}.$$

Proof From Theorem 3.16, the global target privacy loss ε_g of a k-fold sequential adaptive composition $\mathcal{M} = (\mathcal{M}_k \circ ... \circ \mathcal{M}_1)$ is expressed as

$$\varepsilon_g = \varepsilon\sqrt{2k \log \frac{1}{\delta'}} + k\varepsilon\frac{e^\varepsilon - 1}{e^\varepsilon + 1}. \tag{A.1}$$

When $\varepsilon \in (0,1)$, the quantity $\frac{e^\varepsilon - 1}{e^\varepsilon + 1}$ is close to $\frac{\varepsilon}{2}$. Therefore, we obtain

$$\varepsilon_g = \varepsilon\sqrt{2k \log \frac{1}{\delta'}} + \frac{k\varepsilon^2}{2}. \tag{A.2}$$

To express the local privacy loss ε of each mechanism $\mathcal{M}_i$ as a function of the global privacy loss ε_g, we require

$$\frac{k\varepsilon}{2} \leq \sqrt{2k \log \frac{1}{\delta'}} \tag{A.3}$$

which holds for $\delta' \leq \frac{1}{e^{\frac{k\varepsilon}{8}}}$. In practice, this upper-bound on δ' is loose enough to benefit from the tightness of the Advanced Composition (Theorem 3.16) over Basic Composition (Theorem 3.14). Equation A.2 then becomes

$$\varepsilon_g = \varepsilon\sqrt{2k \log \frac{1}{\delta'}} + \varepsilon\sqrt{2k \log \frac{1}{\delta'}} \tag{A.4}$$

$$= 2\varepsilon\sqrt{2k \log \frac{1}{\delta'}}. \tag{A.5}$$

□

A.3 Privacy Amplification by Subsampling

Lemma A.3 (Privacy Amplification by Subsampling) *Suppose $\varepsilon \in (0,1)$, a database D, a sample $S \subseteq D$ including each record of D with probability $q \in (0,1)$, and an (ε,δ)-differentially private mechanism $\mathcal{M}$, then $\mathcal{M}(S)$ is $(2q\varepsilon, q\delta)$-differentially private.*

Proof Let $D, D' \in \mathcal{D}$ be two adjacent databases such that $D = D' \cup \{x\}$, $S \subseteq D$ be an arbitrary sample with sampling probability q, and $\mathcal{M}\colon \mathcal{D} \to \mathcal{R}$ be an (ε,δ)-differentially private mechanism with $\varepsilon \in (0,1)$. Fix a set of output results $E \subseteq \mathcal{R}$. Note that

$$\Pr(\mathcal{M}(D) \in E \mid x \notin S) = \Pr(\mathcal{M}(D') \in E), \tag{A.6}$$
$$\Pr(\mathcal{M}(D) \in E \mid x \in S) \leq e^{\varepsilon}\Pr(\mathcal{M}(D') \in E) + \delta, \tag{A.7}$$

where Equation A.7 directly follows from the definition of approximate DP. Then,

$$\begin{aligned}
\Pr(\mathcal{M}(D) \in E) &= (1-q)\Pr(\mathcal{M}(D) \in E \mid x \notin S) + q\Pr(\mathcal{M}(D) \in E \mid x \in S) \\
&= (1-q)\Pr(\mathcal{M}(D') \in E) + q\Pr(\mathcal{M}(D) \in E \mid x \in S) \\
&\leq (1-q)\Pr(\mathcal{M}(D') \in E) + qe^{\varepsilon}\Pr(\mathcal{M}(D') \in E) + q\delta \\
&= [1 + q(e^{\varepsilon} - 1)]\Pr(\mathcal{M}(D') \in E) + q\delta \\
&\leq (1 + 2q\varepsilon)\Pr(\mathcal{M}(D') \in E) + q\delta \\
&\leq e^{2q\varepsilon}\Pr(\mathcal{M}(D') \in E) + q\delta.
\end{aligned}$$

□

A.4 Encrypted Neural Network Benchmark

This section provides benchmark timings for encrypted training of a 3-layer fully-connected NN with a single hidden layer of 92 neurons on the MNIST dataset using the CKKS-based MHE scheme presented in Section 2.2.3. Our experimental setup uses the sigmoidal linear unit (SiLU) as activation function, a batch of 256 samples, and the MSE loss function. We approximate the SiLU activation function and its derivative by a polynomial of degree 63 in the interval $[-16, 16]$. Concerning CKKS parameters, we use a precision of 24 bits, a ring degree $N = 2^{16}$, and an initial level $L = 23$.

Table A.1 provides the execution time of matrix operations involved in the forward and backward passes in NN training introduced in Section 2.1.2.

Operations	**Execution Time** $[sec]$
Matrix Multiplication	0.74
SiLU Activation	0.42
Collective Bootstrapping	1.6

Table A.1: Benchmark of matrix operations in forward and backward passes. By Jean-Philippe Bossuat, Tune Insight SA, 2023.

Figure A.1 depicts the different matrix operations involved in the encrypted training of a 3-layer NN using the CKKS-based MHE scheme.

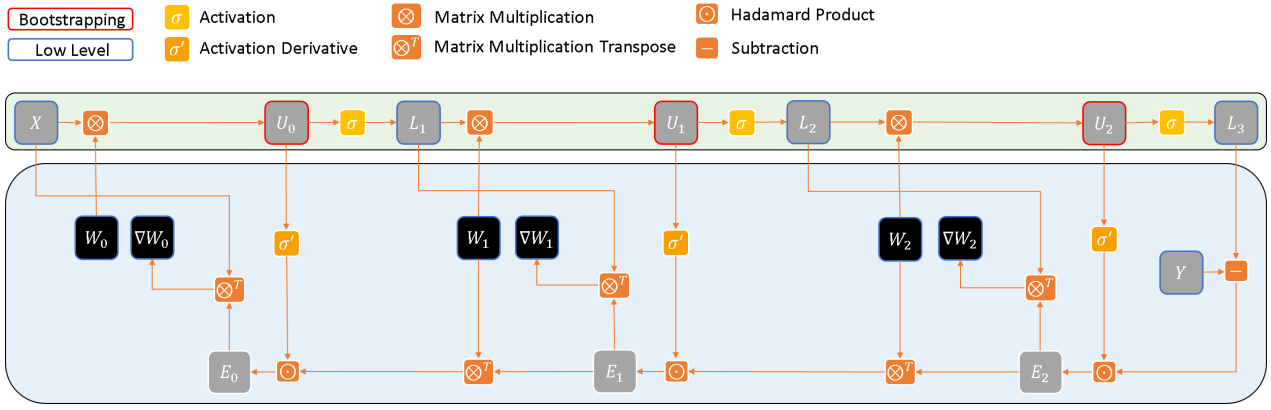


Figure A.1: 3-Layer Fully-Connected NN. Based on material from Jean-Philippe Bossuat, Tune Insight SA, 2023.

Using Table A.1, we derive the total execution time of the forward and backward passes of a 3-layer fully-connected NN on the MNIST dataset using the CKKS-based MHE scheme for a batch of 256 samples in Table A.2.

On average, we assume a single collective bootstrapping operation required in the backward pass.

	Execution Time $[sec]$	
Operations	*Forward Pass*	*Backward Pass*
Matrix Multiplication	$3 \cdot 0.74$	$5 \cdot 0.74$
SiLU Activation	$3 \cdot 0.42$	$3 \cdot 0.42$
Collective Bootstrapping	$3 \cdot 1.6$	1.6
Total	$\underbrace{8.28 \qquad 6.56}_{14.84}$	

Table A.2: Benchmark of a 3-layer fully-connected NN with a single hidden layer of 92 neurons on the MNIST dataset using the CKKS-based MHE scheme for a batch of 256 samples.

Bibliography

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, page 308–318. Association for Computing Machinery, 2016.

[2] Amazon Web Services (AWS). Amazon Elastic Compute Cloud (EC2). https://aws.amazon.com/ec2/.

[3] D. Balson and W. Dixon. Cyber information sharing: Building collective security. In *2020 Annual Report*. World Economic Forum, October 2020.

[4] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, page 1175–1191. Association for Computing Machinery, 2017.

[5] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology*, page 868–886, 2012.

[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology*. Association for Computing Machinery, 2014.

[7] P. Bukaty. *The California Privacy Rights Act (CPRA) – An implementation and compliance guide*. IT Governance Publishing, 2021.

[8] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. Cryptology ePrint Archive, 2016.

[9] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology - ASIACRYPT 2017*, pages 409–437, 2017.

[10] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, page 1655–1658. Association for Computing Machinery, 2018.

[11] J. Deng, R. Socher, L. Fei-Fei, W. Dong, K. Li, and L.-J. Li. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[12] A. Donovan and B. Kernighan. The Go programming language. https://golang.org/.

[13] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *International Conference on the Theory and Application of Cryptographic Techniques*, 2006.

[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography*, pages 265–284, 2006.

[15] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Now Publishers Inc., 2014.

[16] C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60, 2010.

[17] D. Enthoven and Z. Al-Ars. An overview of federated deep learning privacy attacks and defensive strategies. *Computing Research Repository*, 2020.

[18] EPFL-LDS. Lattigo. https://github.com/tuneinsight/lattigo, August 2022. Maintained by Tune Insight SA.

[19] European Commission. General Data Protection Regulation (GDPR), 2016.

[20] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, page 144, 2012.

[21] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, page 1322–1333. Association for Computing Machinery, 2015.

[22] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Standford University, 2009.

[23] R. C. Geyer, T. J. Klein, and M. Nabi. Differentially private federated learning: A client level perspective, 2019.

[24] Government of Singapore. Personal Data Protection Act (PDPA), 2012.

[25] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, 2019.

[26] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium*, pages 1895–1912, 2019.

[27] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 2017.

[28] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009.

[29] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[30] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *40th IEEE Symposium on Security and Privacy*, pages 691–706, 2019.

[31] I. Mironov. Rényi differential privacy. In *IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, 2017.

[32] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, pages 619–640, 2021.

[33] C. Mouchet, J. Troncoso-Pastoriza, J.-P. Bossuat, and J.-P. Hubaux. Multiparty homomorphic encryption from Ring-Learning-With-Errors. *Proceedings on Privacy Enhancing Technologies*, pages 291–311, 2021.

[34] J. Murtagh and S. Vadhan. The complexity of computing the optimal composition of differential privacy. *Theory of Computing*, 14, 2018.

[35] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy*, pages 739–753, 2019.

[36] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, page 223–238, 1999.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library, 2019.

[38] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, pages 1333–1345, 2018.

[39] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[40] A. Sannai. Reconstruction of training samples from loss functions. *ArXiv*, 2018.

[41] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux. POSEIDON: Privacy-preserving federated neural network learning. *28th Annual Network And Distributed System Security Symposium*, 2021.

[42] A. Smith and J. Ullman. Privacy in statistics and machine learning - Lecture 10: Advanced composition, 2021.

[43] S. Song, K. Chaudhuri, and A. Sarwate. Stochastic gradient descent with differentially private updates. *Proceedings of the IEEE Global Conference on Signal and Information Processing*, pages 245–248, 2013.

[44] M. Tan and Q. V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114, 2019.

[45] J. R. Troncoso-Pastoriza, A. Mermoud, R. Bouyé, F. Marino, J.-P. Bossuat, V. Lenders, and J.-P. Hubaux. Orchestrating collaborative cybersecurity: A secure framework for distributed privacy-preserving threat intelligence sharing. *arXiv*, 2022.

[46] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, page 1–11. Association for Computing Machinery, 2019.

[47] S. Truex, L. Liu, K. H. Chow, M. E. Gursoy, and W. Wei. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. Association for Computing Machinery, 2020.

[48] N. Truong, K. Sun, S. Wang, F. Guitton, and Y. Guo. Privacy preservation in federated learning: An insightful survey from the GDPR perspective. *Computers and Security*, 110, 2021.

[49] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. *IEEE Conference on Computer Communications*, pages 2512–2520, 2018.

[50] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, and H. V. Poor. User-level privacy-preserving federated learning: Analysis and performance optimization. *IEEE Transactions on Mobile Computing*, pages 3388–3401, 2021.

[51] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, page 160–164. IEEE Computer Society, 1982.

[52] A. C.-C. Yao. How to generate and exchange secrets. *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[53] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 Conference on Computer and Communications Security*, page 3093–3106. Association for Computing Machinery, 2022.

[54] X. Yin, Y. Zhu, and J. Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys*, 2021.

[55] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, 2019.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.