University of Brasília

Institute of Exact Sciences
Department of Computer Science

P^2MLF: An End-to-End Privacy-Preserving Framework for Machine Learning Applications

Ricardo José Menezes Maia

Document submitted in partial fullfilment of the requirements to Doctoral Degree in Informatics

Advisor
Prof. Dr. Ricardo Pezzuol Jacobi

Co-advisor
Prof. Dr. Anderson Clayton Alves Nascimento

Brasília
2024

University of Brasília

Institute of Exact Sciences
Department of Computer Science

P^2MLF: An End-to-End Privacy-Preserving Framework for Machine Learning Applications

Ricardo José Menezes Maia

Document submitted in partial fullfilment of the requirements to Doctoral Degree in Informatics

Brasília, November 28th 2024

Dedications

I dedicate this work, first and foremost, to God, who has guided my actions throughout my life. My mother and father have always been by my side. To my brothers and sister, for their unwavering support, and to my beloved wife, whose love and encouragement sustained me at every moment. I also dedicate this work to the friends who, in one way or another, accompanied me on this journey, offering their support and understanding of the time, energy, and dedication it demanded.

7 Getting wisdom is the wisest thing you can do!
And whatever else you do, develop good judgment.
8 If you prize wisdom, she will make you great.
Embrace her, and she will honor you.

Proverbs 4:7-8, The Holy Bible.

Acknowledgements

First, I am thankful to God for everything. I include my mother, father, brothers, and sister, whose constant love and support have given me the energy to pursue the purposes I believe in.

I owe my wife thanks for her unwavering support and belief in me throughout this journey of seeking knowledge. She has been by my side, sharing the challenges and triumphs of this research. Her patience and willingness to listen to my practice sessions countless times have made her as familiar with this work as I am.

I am grateful to the Danish Data Science Academy program for the opportunity to visit the IT University of Copenhagen and INESC TEC, which allowed me to serve as a visiting researcher at the University of Porto.

I am grateful to the PPML Huskies team at the University of Washington Tacoma. The lessons I have learned from this group have been invaluable. It was an honor to win second place in the competition organized by the US White House and first place in iDASH 2021, but above all, to experience the true spirit of inclusive and collaborative teamwork. These achievements, which I never imagined possible, were made even more special by the warm welcome and camaraderie of the friends I made in Tacoma and Seattle.

I am grateful to my co-advisor and friend for introducing me to the fascinating world of modern cryptography, showing me how to combine it with artificial intelligence, and challenging me to expand my horizons. I am eternally grateful to my advisor for opening the door that allowed me to pursue a PhD at the University of Brasília and for his invaluable guidance throughout my research.

There are countless others to whom I owe immense gratitude—friends, colleagues, and mainly mentors—who have inspired me to persevere in one way or another. To the friends I met on this journey at the University of Brasília, the University of Washington Tacoma, the University of IT Copenhagen, and the University of Porto, meeting you in person and sharing this journey was an unforgettable experience.

Finally, I am grateful to the Brazilian public university system, which provided me with my undergraduate, master's, and now doctoral degrees, making all of this possible.

Resumo

P^{2}MLF: Um Arcabouço para Preservação de Privacidade de Ponta a Ponta para Aplicações de Aprendizado de Máquina

O direito à privacidade de dados é fundamental para indivíduos e empresas. Pode-se mencionar os benefícios dos aplicativos Machine Learning (ML) para pessoas e empresas. Por isso, é essencial encontrar soluções que garantam a privacidade de dados em aplicações que utilizam ML, especialmente em cenários onde os aplicativos ML têm requisitos de privacidade não funcionais por razões legais.

No requisito de privacidade abordado neste trabalho, apenas o proprietário terá acesso aos seus dados. Problemas de privacidade podem surgir nos estágios de entrada e saída da aplicação de ML e, para ilustrar esse problema, considere *Alice* como o proprietário das informações e *Bob* como o proprietário do modelo ML.Garantir a privacidade de entrada significa impedir a exposição dos dados, preservando tanto a privacidade das informações de *Alice* quanto a propriedade intelectual do modelo de *Bob*. Garantir a privacidade de saída significa que *Bob* não precisa expor seu modelo em texto simples para *Alice*, e *Alice* não precisa revelar seus dados em texto simples para o modelo de *Bob*. Mesmo com privacidade de entrada, *Alice* poderia potencialmente explorar informações usadas por *Bob* no treinamento do modelo, e a privacidade de saída evita vazamento de dados durante o treinamento. Garantir a privacidade dos dados de entrada e saída durante a inferência e o treinamento é essencial para a proteção de privacidade de ponta a ponta em aplicativos ML.

Visando solucionar o problema descrito, este trabalho tem como objetivo principal propor uma abordagem para garantir privacidade de ponta a ponta, abrangendo tanto as entradas quanto as saídas, em aplicativos de ML, denominada Privacy-Preserving Machine Learning Framework (P^{2}MLF).

Entre os objetivos secundários desta tese, destaca-se a demonstração da aplicabilidade de P^{2}MLF, utilizando uma aplicação baseada em Secure Multi-Party Computation (MPC) para inferência segura de Malicious Software (Malware) usando modelos MultiLayer Perceptron

(MLP), One-Dimensional Convolutional Neural Network (CNN1D) e Long Short-Term Memory (LSTM) treinados com Differentially-Private Stochastic Gradient Descent (DP-SGD). Outra aplicação utilizará os métodos de P^{2}MLF para treinamento colaborativo de modelos Collaborative Intrusion Detection Systems (CIDS).

Uma das contribuições do método de inferência do P^{2}MLF é o uso de quantização float16 pós-treinamento de modelos de aprendizado profundo com MPC para obter detecção eficiente e segura de Domain Generation Algorithms (DGA). Este trabalho demonstra que a quantização aumenta significativamente a velocidade, reduzindo o tempo de execução da inferência em 23% a 42%, sem prejuízo à precisão, utilizando um protocolo de computação segura de três partes. Soluções anteriores não garantem privacidade de ponta a ponta, não fornecem garantias de Differential Privacy (DP) para resultados do modelo e assumem que os Embedding Layer (EL)s do modelo são conhecidos publicamente. O melhor protocolo em termos de precisão é executado em aproximadamente $0,22$ segundos.

Por fim, a segunda contribuição destaca a avaliação dos três métodos de treinamento colaborativo propostos pelo P^{2}MLF, com foco em escalabilidade e privacidade, aplicados ao treinamento do CIDS. Entre os métodos avaliados, o que demonstrou o melhor equilíbrio entre privacidade e escalabilidade foi aquele que combina um protocolo MPC para agregação com modelos locais diferencialmente privados, treinados por meio de aprendizagem federada. Esse método é aproximadamente 1,5 vezes mais rápido que a abordagem de maior privacidade, que utiliza exclusivamente protocolos MPC com garantias de DP.

Palavras-chave: Computação Multipartidária Segura, Privacidade Diferencial, Aprendizagem Federada, Detecção de Algoritmos de Geração de Domínios, Sistema de Detecção de Intrusão Colaborativo, Preservação de Privacidade em Aprendizagem de Máquina

Abstract

The right to data privacy is fundamental for individuals and companies. One can mention the benefits of Machine Learning (ML) applications for people and businesses. Therefore, finding solutions to balance the dilemma of ensuring data privacy in applications that use ML is vital, especially in scenarios where ML applications have non-functional privacy requirements for legal reasons.

In the privacy requirement addressed in this work, only the data owner will know their data. Privacy issues can arise in the input and output stages of the application of ML, and to illustrate this problem, consider *Alice* as the owner of the information and *Bob* as the owner of the model ML. Ensuring input privacy means that data should not be exposed to avoid compromising the privacy of *Alice* 's data or the intellectual property of *Bob* 's model. Ensuring output privacy means that *Bob* does not need to expose his model in plain text to *Alice*, and *Alice* does not need to reveal her data in plain text to *Bob* 's model. Even with input privacy, *Alice* could potentially exploit information used by *Bob* in model training, and output privacy prevents data leakage during training. Ensuring the privacy of the input and output data during inference and training is essential for end-to-end privacy protection in ML applications.

Concerning solving the problem proposed, this work's main objective is to propose an approach to ensure end-to-end privacy, encompassing inputs and outputs, in ML applications, referred to as Privacy-Preserving Machine Learning Framework (P^{2}MLF).

This thesis will demonstrate as secondary objectives the framework's applicability through an application that uses Secure Multi-Party Computation (MPC) for private inference of Malicious Software (Malware), using MultiLayer Perceptron (MLP), One-Dimensional Convolutional Neural Network (CNN1D), and Long Short-Term Memory (LSTM) models trained with Differentially-Private Stochastic Gradient Descent (DP-SGD). Another application will apply the methods described in this work for collaborative training Collaborative Intrusion Detection Systems (CIDS) models.

In addition, to mention one of the contributions of P^{2}MLF inference method, this work uses post-training float16 quantization of deep learning models with MPC to achieve efficient and secure detection of Domain Generation Algorithms (DGA). This work demonstrates that quantization significantly increases speed, resulting in a 23% to 42% reduction in inference execution time without reducing accuracy, using a three-party secure computation protocol that tolerates one corruption. Previous solutions are not end-to-end private, do not provide Differential Privacy (DP) guarantees for model results, and assume that the model's Embedding Layer (EL)s are publicly known. The best protocol in terms of accuracy runs in approximately 0.22 seconds.

Finally, as a second contribution, it emphasizes evaluating the three collaborative training methods proposed by P^{2}MLF, focusing on scalability and privacy, applied to the training of CIDS. Among the evaluated methods, the one that presented the best balance between privacy and scalability - being 1.50 times faster than the approach with the highest privacy, based exclusively on MPC protocols with DP guarantees — is the method that combines an MPC protocol for aggregation of local models with DP guarantees and trained through Federated Learning (FL).

Keywords: Secure Multi-party Computation, Differential Privacy, Federated Learning, Domain Generation Algorithms Detection, Collaborative Intrusion Detection Systems, Privacy-Preserving Machine Learning

Resumo Extendido

P^{2}MLF: Um Arcabouço para Preservação de Privacidade de Ponta a Ponta para Aplicações de Aprendizado de Máquina

Para atender às normas da Universidade de Brasília, este capítulo apresenta um resumo de cada capítulo em Português.

Capítulo 1 - Introdução

Esta tese se concentra em técnicas para garantir a privacidade dos dados durante as fases de treinamento e inferência de modelos de Machine Learning (ML). Este trabalho utiliza estudos de caso práticos para demonstrar a eficácia dos métodos propostos em Privacy-Preserving Machine Learning Framework (P^{2}MLF) para preservar a privacidade das informações enquanto viabilizam operações de ML.

ML usa dados para generalizar e aprender padrões sem definir explicitamente esses comportamentos por meio de programação. Os modelos ML dependem de grandes quantidades de dados para aprender um amplo espectro de padrões [1].

Esses dados podem ser obtidos de diversas origens, como dispositivos Internet of Things (IoT) [2], dados de rede de computadores [3, 4], usuários, entidades governamentais, dados hospitalares [5], dados cerebrais [6, 7], destacando o amplo escopo do campo.

Embora muitas aplicações ML benéficas estejam surgindo, há uma crescente preocupação social sobre a privacidade de dados sensíveis utilizados durante o treinamento e a inferência de modelos ML [8].

Um exemplo de uma aplicação ML com requisitos de privacidade é a detecção de crimes financeiros usando dados de diferentes instituições financeiras em vários países [9]. Outro exemplo envolve o treinamento de modelos ML com dados de diferentes hospitais para detectar doenças cardíacas [5], câncer ou COVID-19 usando raios X de tórax e dados clínicos. Outros exemplos incluem sistemas de recomendação que usam dados confidenciais do usuário, como orientação sexual, religiosa ou política. Na segurança cibernética, surgem desafios quando uma empresa deseja usar um sistema de detecção de intrusão [3], mas deve proteger dados confidenciais, como propriedade industrial ou segredos comerciais, devido às leis de privacidade.

Dados sensíveis incluem informações como origem racial ou étnica, crenças religiosas, opiniões políticas, filiações sindicais e dados relacionados à saúde ou vida sexual [10]. Também englobam dados genéticos, renda, empréstimos, informações fiscais e biométricas vinculadas a uma pessoa física [10]. Para empresas, dados sensíveis podem incluir informações do cliente ou detalhes sobre faturamento, lucros e salários.

Em alguns cenários, é possível remover dados sensíveis. No entanto, essa remoção pode ser impraticável quando essas informações são essenciais para melhorar o desempenho do modelo durante o treinamento ou necessárias para a inferência. Portanto, é fundamental desenvolver métodos que garantam a privacidade, especialmente quando os algoritmos de ML utilizam dados sensíveis [11–13]. A área de pesquisa que concilia ML com privacidade é conhecida como Privacy-Preserving Machine Learning (PPML). Essa área é crucial porque dados sensíveis geralmente são úteis em uma infinidade de aplicações e podem ser essenciais na criação de modelos ML benéficos em vários domínios, como medicina, finanças ou até mesmo na segurança cibernética.

A motivação para abordar esse problema decorre de discussões sociais sobre privacidade [14], uma vez que o direito à privacidade dos dados é uma das garantias fundamentais do indivíduo. Em uma sociedade orientada à informação, muitos dados confidenciais são coletados sobre indivíduos e empresas. Portanto, o problema de garantir privacidade de dados e, ao mesmo tempo, possibilitar o seu uso em modelos de ML pode viabilizar o desenvolvimento de inúmeras aplicações que beneficiam os indivíduos. Por exemplo, informações pessoais em redes sociais podem ser usadas para fazer recomendações personalizadas de notícias, livros, roupas, destinos de viagem, hotéis, restaurantes, filmes e música [11–13].

Esta tese propõe o arcabouço P^{2}MLF, que integra protocolos de Secure Multi-Party Computation (MPC) para inferência e modelos treinados com Differential Privacy (DP), sejam eles colaborativos ou não colaborativos. Em síntese, o foco do arcabouço propoto é garantir a privacidade de dados em algoritmos de ML [15], com ênfase especial no próprio P^{2}MLF. O trabalho apresenta estudos de caso aplicados à inferência de Malicious Software (Malware) e ao treinamento colaborativo de modelos ML para detecção de intrusão. O arcabouço proposto garante a privacidade de dados durante as fases de treinamento e inferência de modelos focados em segurança cibernética. Para inferência segura, este trabalho propõe usar protocolos MPC e modelos treinados com DP. Para o treinamento, este trabalho propõe métodos colaborativos e não colaborativos com diferentes níveis de privacidade, segurança e desempenho. O primeiro método é colaborativo e combina Federated Learning (FL) com DP; o segundo é colaborativo e combina MPC com DP; o terceiro é colaborativo e utiliza FL com DP e agregação com MPC para garantir a privacidade dos dados; e o quarto método é não colaborativo e utiliza apenas DP. A proposta de diferentes métodos de treinamento busca discutir o uso ideal de cada abordagem, considerando os níveis de privacidade, desempenho e a presença ou ausência de colaboração.

Problema

Problemas de privacidade surgem nos estágios de entrada e saída em aplicações que utilizam ML. Para ilustrar esses problemas, considere que *Alice* é a proprietária das informações e *Bob* é o proprietário do modelo. No estágio de entrada, os dados usados para detecção podem ser expostos, comprometendo a privacidade dos dados de *Alice* ou a propriedade intelectual do modelo de *Bob*. Garantir a privacidade da entrada significa que *Bob* não expõe seu modelo para *Alice*, assim como *Alice* não expõe seus dados para o modelo de *Bob*. Da mesma forma, no estágio de saída, os resultados da classificação podem revelar informações sobre o modelo, permitindo sua exploração para realizar engenharia reversa dos dados de treinamento. Garantir a privacidade dos dados de entrada e saída durante a inferência e o treinamento é essencial para a proteção de privacidade de ponta a ponta.

Portanto, o principal desafio é propor um arcabouço computacional que garanta a privacidade de entrada e saída em aplicações de aprendizagem de máquina.

Motivação para o Problema

O direito à privacidade de dados pessoais é um princípio fundamental garantido por legislações específicas [10,16], essencial para a proteção dos indivíduos. Ao mesmo tempo, os benefícios das aplicações que utilizam ML estão se tornando amplamente difundidos na sociedade, de forma que buscar soluções para resolver o problema de combinar ML e privacidade é tarefa extremamente relevante.

Para atingir esse propósito, é necessário superar o dilema de utilizar dados sensíveis sem que cada parte tenha acesso às informações confidenciais das demais partes envolvidas nas fases de treinamento e inferência dos modelos ML. Embora os dados sensíveis possam, teoricamente, ser armazenados com segurança e nunca usados, essa abordagem é impraticável, especialmente quando os dados podem melhorar a precisão dos algoritmos de aprendizado de máquina ou são essenciais em determinadas aplicações.

O desafio está em encontrar maneiras de controlar o vazamento de dados sensíveis enquanto eles são armazenados, comunicados ou processados, mesmo quando o proprietário dos dados não confia nas outras partes envolvidas. Esse problema é particularmente relevante, pois diversos serviços valiosos de ML podem ser desenvolvidos usando dados sensíveis de diferentes fontes. Por exemplo, dados confidenciais de vários indivíduos podem ser combinados para detectar pandemias ou doenças. Outro exemplo é a detecção de crimes relacionados à pornografia infantil, garantindo simultaneamente a proteção da privacidade dos dados dos suspeitos [17].

O problema torna-se desafiador devido à sobrecarga computacional introduzida pela adoção de protocolos criptográficos como MPC o que leva ao aumento do tempo de execução, de memória e de consumo de energia. Além disso, técnicas como DP podem comprometer as métricas de ML, especialmente à medida que o nível de privacidade aumenta.

Para ilustrar a importância de garantir a privacidade na entrada e na saída, consideram-se as fases de inferência e treinamento. Na fase de inferência em ML, a privacidade de entrada é essencial para proteger tanto os dados de *Alice* quanto o modelo de *Bob*. Durante o treinamento, a privacidade de entrada viabiliza a criação de um modelo global de ML que preserva a privacidade dos dados de diferentes partes, como *Alice*, *Bob*, *Clara* e *Dan*. Já a privacidade de saída impede que *Alice*, durante a inferência, obtenha informações sobre os dados de treinamento utilizados no modelo de *Bob*.

Objetivos

Esta tese apresenta o arcabouço P^{2}MLF, projetado para garantir a privacidade tanto na entrada quanto na saída de modelos de aprendizado de máquina, otimizando simultaneamente a eficiência no uso das tecnologias MPC, FL e DP. O objetivo principal é reduzir o tempo de execução e as rodadas de comunicação em rede, mantendo a acurácia dos modelos treinados em níveis comparáveis aos daqueles desenvolvidos sem mecanismos de privacidade.

Além do objetivo principal, este trabalho busca atingir os seguintes objetivos secundários:

- Aplicar o arcabouço P^{2}MLF à inferência com privacidade em Malware, utilizando protocolos MPC para modelos MultiLayer Perceptron (MLP), One-Dimensional Convolutional Neural Network (CNN1D) e Long Short-Term Memory (LSTM). Este trabalho busca identificar o modelo ML que oferece o melhor equilíbrio entre privacidade, menor tempo de execução e menores taxas de erro dentre os três protocolos MPC implementados.
- Demonstrar a aplicação do P^{2}MLF no treinamento colaborativo, utilizando como estudo de caso um sistema de detecção de intrusão treinado de forma colaborativa.

Resultados e Contribuições Científicas

Portanto, este trabalho propõe o arcabouço P^{2}MLF, aplicado à detecção automatizada e terceirizada de Domain Generation Algorithms (DGA), garantindo a privacidade tanto dos dados dos usuários da rede empresarial quanto dos provedores de serviços de detecção de DGA. Nesse arcabouço, o tráfego de Domain Name System (DNS) não é exposto em texto claro aos provedores de serviços, e o modelo não é compartilhado com os administradores da rede, assegurando que nenhuma informação sensível seja revelada a qualquer outra parte. Além disso, o trabalho incorpora técnicas para evitar que invasores obtenham informações adicionais sobre os dados de treinamento ou reconstruam o modelo a partir dos resultados de classificação enviados aos administradores da rede.

O trabalho mais relacionado à detecção com preservação de privacidade de DGAs é apresentado em [18]. Esse trabalho fornece privacidade apenas na entrada, utilizando MPC, mas deixa a fase de inferência vulnerável a ataques que exploram os dados de treinamento (consulte o capítulo 3 para mais detalhes). Em contraste, esta tese propõe o primeiro arcabouço de preservação de privacidade de ponta a ponta para classificação de DGAs, garantindo tanto a privacidade de entrada quanto de saída. As contribuições podem ser resumidas a seguir:

- Este trabalho propõe o arcabouço P^{2}MLF para oferecer classificação segura como um serviço para domínios DGA/não-DGA, garantindo privacidade de entrada (MPC) e privacidade de saída (DP).
- O arcabouço proposto, P^{2}MLF, é o primeiro a considerar o treinamento diferencialmente privado de modelos para a classificação de DGA.
- O P^{2}MLF suporta classificadores baseados em MLP, CNN1D e LSTM. O protocolo MPC desenvolvido para LSTM é inovador, eficiente e representa a primeira implementação desse tipo no arcabouço computacional MPC chamado MP-SPDZ [19].
- Este trabalho avalia o P^{2}MLF utilizando conjuntos de dados reais — DGArchive e Alexa — em tarefas de classificação de nomes de domínio binários e multiclasse. No problema de classificação binária, a tarefa é distinguir se um domínio é benigno ou malicioso. Já no problema de classificação multiclasse, a tarefa inclui identificar a família DGA correspondente ao domínio malicioso.
- Este trabalho analisa empiricamente as compensações entre privacidade e utilidade do P^{2}MLF ao utilizar os modelos MLP, CNN1D e LSTM. Os resultados mostram que a privacidade de saída, garantida pela introdução de ruído para atender às exigências de DP, degrada a precisão apenas de forma mínima nos experimentos. Por outro lado, o uso de protocolos MPC para garantir a privacidade de entrada não compromete a utilidade do modelo de classificação. O modelo mais rápido em termos de tempo de execução é o MLP, com um tempo de inferência de $0,07$ segundos. Enquanto que o modelo com a melhor precisão é o CNN1D, que alcança uma precisão de 93% com um epsilon igual a 5.
- Este trabalho demonstra a eficiência do P^{2}MLF em termos de tempo de execução, considerando cenários com 2 ou 3 partes de computação.
- Este trabalho observa que a quantização proporciona melhorias significativas no desempenho dos protocolos MPC, reduzindo as rodadas de comunicação e o tempo de inferência. Os experimentos indicam uma redução de 23% a 42% no tempo de execução da inferência, sem comprometer a precisão, na configuração com 3 partes (usando replicated secret sharing). Os resultados demonstram que é possível alcançar detecção segura quase em tempo real de domínios DGA.
- Uma das principais contribuições do treinamento de Collaborative Intrusion Detection Systems (CIDS) utilizando o P^{2}MLF é a avaliação de três métodos de treinamento colaborativo propostos, com foco no equilíbrio entre escalabilidade e privacidade. Dentre os métodos analisados, o que demonstrou o melhor equilíbrio foi aquele que combina um protocolo MPC para agregação de dados com modelos locais diferencialmente privados, treinados por meio de aprendizagem federada. Esse método se mostrou aproximadamente $1,5$ vezes mais rápido do que a abordagem mais focada em privacidade, que utiliza exclusivamente protocolos MPC com garantias de DP.

Contents

1 Introduction **1**

1.1 Problem . . . 3

1.2 Motivation for the problem . . . 3

1.3 Objectives . . . 4

1.4 Results and Scientific Contributions . . . 4

1.5 Publications and Competitions . . . 5

1.6 Outline of the Thesis . . . 7

2 Background **8**

2.1 Privacy-Enhancing Technologies . . . 8

2.1.1 Secure Multi-Party Computation . . . 8

2.1.1.1 History of MPC . . . 9

2.1.1.2 Applications of MPC . . . 10

2.1.1.3 Basic primitives . . . 11

2.1.1.4 Secret Sharing . . . 11

2.1.1.5 Protocol for addition . . . 12

2.1.1.6 Protocol for addition by a constant . . . 12

2.1.1.7 Protocol for multiplication by a scalar . . . 12

2.1.1.8 Protocol for Multiplication . . . 12

2.1.1.9 Adversarial Models . . . 13

2.1.1.10 Real-World and Ideal-World Paradigms . . . 13

2.1.1.11 Semi-honest or curious . . . 14

2.1.1.12 Malicious . . . 15

2.1.1.13 The need for a trusted third party . . . 15

2.1.1.14 Outsourced Computation . . . 16

2.1.2 Universal Composability . . . 16

2.1.3 Differential Privacy . . . 16

2.1.3.1 Differentially-Private Stochastic Gradient Descent . . . 17

2.1.4 Federated Learning . . . 18

2.2 Machine Learning and Deep Learning 18
2.2.1 Logistic Regression 19
2.2.2 Multilayer Perceptron 19
2.2.3 Long Short-Term Memory Networks 20
2.2.4 Convolutional Neural Network 21
2.2.5 Preprocessing 22
2.2.5.1 Embedding layer 22
2.2.6 Post-Processing 23
2.2.6.1 Quantization 23
2.3 Applications 23
2.3.1 Domain Generation Algorithms 23
2.3.2 Intrusion Detection System 25

3 Related Works 27
3.1 DGA detection using deep learning 27
3.2 Secure Multi-Party Computation for DGA Detection 29
3.3 Secure Multi-Party Computation for Natural Language Processing 30
3.4 IDS with Privacy-Preserving Machine Learning 31
3.5 Attacks and Defenses on Machine Learning 32
3.6 Frameworks PPML 33

4 P^{2}MLF 35
4.1 Introduction 35
4.2 Threat Model 38
4.3 Privacy Requirements 38
4.4 Basic Building Blocks 38
4.5 P^{2}MLF Preprocessing 39
4.5.1 Embeddings 39
4.6 P^{2}MLF Models 40
4.6.1 MLP 41
4.6.2 CNN1D 41
4.6.3 LSTM 42
4.6.4 LR 42
4.7 P^{2}MLF Inference 43
4.8 P^{2}MLF Training 43
4.8.1 Basic Building Blocks 44
4.8.2 Collaborative Training 44
4.8.2.1 DP with FL 45

4.8.2.2 DP with FL(Aggregation on MPC) 46
4.8.2.3 DP with MPC 48
4.8.3 Non-Collaborative Training 50
4.8.3.1 DP (DP-SGD) 50
4.9 Security and Privacy 51
4.9.1 Input privacy 52
4.9.2 Output privacy 52

5 P^{2}MLF Training Applied in CIDS 53
5.1 Introduction 53
5.2 Problem 54
5.3 Motivation to the Problem 56
5.4 Contributions 57
5.5 P^{2}MLF Applied in CIDS 58
5.6 CIDS using P^{2}MLF Collaborative Training 58
5.6.1 P^{2}MLF Collaborative Training DP with FL 58
5.6.2 DP with FL (Aggregation on MPC) 58
5.6.3 P^{2}MLF training MPC with DP 59
5.7 P^{2}MLF Inference of CIDS 59
5.8 Dataset 60
5.9 Utility-Privacy Trade-Off 60
5.10 Security and Privacy 61
5.11 Runtimes 61
5.12 Final considerations 62

6 P^{2}MLF Inference Applied in DGA 64
6.1 Introduction 64
6.2 Problem 65
6.3 Contributions 66
6.4 P^{2}MLF Applied in DGA Detection 66
6.5 DGA Classifiers using P^{2}MLF Non-Collaborative Training 68
6.6 P^{2}MLF inference of DGA domains 69
6.7 Dataset 69
6.8 Model Architectures and Parameters 70
6.9 Utility-Privacy Trade-Off 71
6.10 Runtimes 72
6.11 Final considerations 73

7 Conclusions and Future Work **75**

References **78**

List of Figures

1.1 Example of applications that may require privacy requirements 2

2.1 Illustration of DGA usage: The botmaster and malware on an infected client generate an identical list of domain names. The botmaster registers one of these generated domains. The malware then attempts to resolve each domain in the list via DNS queries until it finds the registered domain, thus establishing a successful connection between the infected client and the CCS server. 25

4.1 P^{2}MLF pipeline 36
4.2 P^{2}MLF Framework Overview 37
4.3 P^{2}MLF Inference 43
4.4 P^{2}MLF Training 44
4.5 P^{2}MLF Collaborative Training using DP with FL 45
4.6 P^{2}MLF Collaborative Training using DP with FL (Aggregation with MPC) 48
4.7 P^{2}MLF Collaborative Training using DP with MPC 49
4.8 P^{2}MLF Non-Collaborative Training with DP 51

5.1 Illustration of benefits and challenges for training a CIDS with data from different parties 55
5.2 Illustration of benefits and challenges of secure inference 55

6.1 The flowchart uses an end-to-end privacy approach to describe the privacy-preserving service for DGA inference. The service provider (*Bob*) trains the DNS domain name classifier with DP-SGD, ensuring DP (output privacy). *Alice* submits new domain names, and *Bob*'s model classifies these domains using MPC. MPC servers execute protocols over encrypted data, maintaining input privacy. 67

List of Tables

5.1 P^{2}MLF Runtimes 62
5.2 Metrics for Different Noise Levels 62
5.3 P^{2}MLF MPC Protocols 62

6.1 Results related to the accuracy of DGA inference were obtained for different noise levels, both with and without the application of quantization [3]. . . 72
6.2 Inference using MPC protocol [3]. 73
6.3 inference using MPC protocol with quantization applied after training with DP [3]. 73

List of Abbreviations

CCS	Command and Control Server.
CIDS	Collaborative Intrusion Detection Systems.
CNN	Convolutional Neural Network.
CNN1D	One-Dimensional Convolutional Neural Network.
DDoS	Distributed Denial of Service.
DGA	Domain Generation Algorithms.
DL	Deep Learning.
DNS	Domain Name System.
DP	Differential Privacy.
DP-SGD	Differentially-Private Stochastic Gradient Descent.
EL	Embedding Layer.
FL	Federated Learning.
HE	Homomorphic Encryption.
IDASH	PRIVACY SECURITY WORKSHOP - Secure Genome Analysis Competition.
IDS	Intrusion Detection System.
IoT	Internet of Things.
LR	Logistic Regression.
LSTM	Long Short-Term Memory.

Malware	Malicious Software.
ML	Machine Learning.
MLP	MultiLayer Perceptron.
MPC	Secure Multi-Party Computation.
NLP	Natural language processing.
P^{2}MLF	Privacy-Preserving Machine Learning Framework.
PET	Privacy-Enhancing Technologies.
PPML	Privacy-Preserving Machine Learning.
RNN	Recurrent Neural Network.
SGD	Stochastic Gradient Descent.
TF-Privacy	TensorFlow Privacy.
UC	Universal Composability.

Symbol Reference

x	A scalar (integer or real), represented by lowercase letters, e.g. x, y.
$\mathbf{x}$	Vector, represented by bold lowercase letters, e.g. $\mathbf{x}$, $\mathbf{y}$.
$\mathbf{X}$	A matrix, represented by bold uppercase letters, e.g. $\mathbf{X}$, $\mathbf{Y}$.
$[\![x]\!]$	A secret sharing of x, where $[\![x]\!]$ is $(x_0, x_1, \cdots, x_n)$
π	Prefix to represent a protocol.
π_{CNN1D}	Secure Multi-Party Computation (MPC) protocol for the One-Dimensional Convolutional Neural Network (CNN1D) layer.
π_{DP}	Add Laplacian Noise Differential Privacy (DP) on the vector.
π_{DENSE}	MPC protocol for the dense layer.
$\pi_{\mathsf{EMBEDDING}}$	MPC protocol for the Embedding Layer (EL) layer.
$\pi_{\mathsf{LRINFERENCE}}$	MPC protocol to secure inference of Logistic Regression (LR) model.
$\pi_{\mathsf{LRTRAINING}}$	MPC protocol to the LR models training with DP guarantees
π_{LSTM}	MPC protocol for the Long Short-Term Memory (LSTM) layer.
π_{MUL}	MPC protocol for dot product.
π_{RELU}	MPC protocol for the relu function.
π_{SIGMOID}	MPC protocol for the sigmoid function.
π_{SOFTMAX}	MPC protocol for the softmax function.
π_{SUM}	MPC protocol for the secure sum.
π_{TANH}	MPC protocol for the hyperbolic tangent function
π_{FLCLIENT}	Federated Learning (FL) client protocol used in the LR models training using DP.
π_{FLSERVER}	FL server protocol to aggregate the models.
$\pi_{\mathsf{SECUREFLCLIENT}}$	FL client used in the LR model training using DP but sending to the server the secret shares of the model.
$\pi_{\mathsf{SECUREFLSERVER}}$	FL server that calls the the protocol $\pi_{\mathsf{FLAGGREGATOR}}$.
$\pi_{\mathsf{FLAGGREGATOR}}$	FL server with MPC protocol to aggregate the models.

Chapter 1

Introduction

This thesis proposes an approach known as Privacy-Preserving Machine Learning Framework (P^2MLF), designed to ensure data privacy throughout the various stages of Machine Learning (ML). P^2MLF is applied to practical scenarios to demonstrate its effectiveness in balancing privacy and scalability in the context of Privacy-Preserving Machine Learning (PPML).

ML uses data to generalize and learn patterns without the need for explicit programming definitions of behaviors. To achieve this, ML models rely on large volumes of data to begin generalizing or learning various patterns [1].

These data can come from various sources, such as Internet of Things (IoT) sensors [2], network data [3, 4], genome, brain data [6, 7], data to detect pandemics, hospitals [5], judicial proceedings, financial institutions [9], and social media, which illustrates the breadth of the field according to figure 1.1. Due to legislation regulating data privacy, privacy guarantees may be necessary for user data, private companies, or the government.

Although the use of ML generates beneficial applications, there is growing societal concern about the privacy of sensitive data used in the training and inference of ML models [8].

An example of a privacy-demanding ML application is the detection of financial crimes using data from different financial institutions in multiple countries [9]. Another example involves using hospital data to perform the training on ML models to detect heart disease [5], cancer, or COVID-19 [20]. Recommendation systems also deal with sensitive user data, such as sexual, religious, or political orientation. In cybersecurity, challenges arise when companies need to adopt intrusion detection systems [3, 4] but must protect sensitive data, such as industrial property or trade secrets, due to privacy laws.

Sensitive data include information such as race, ethnicity, religious beliefs, political opinions, trade union membership, health or sex life data, genetic data, income, loans, tax, and biometric information linked to an individual. For companies, sensitive data

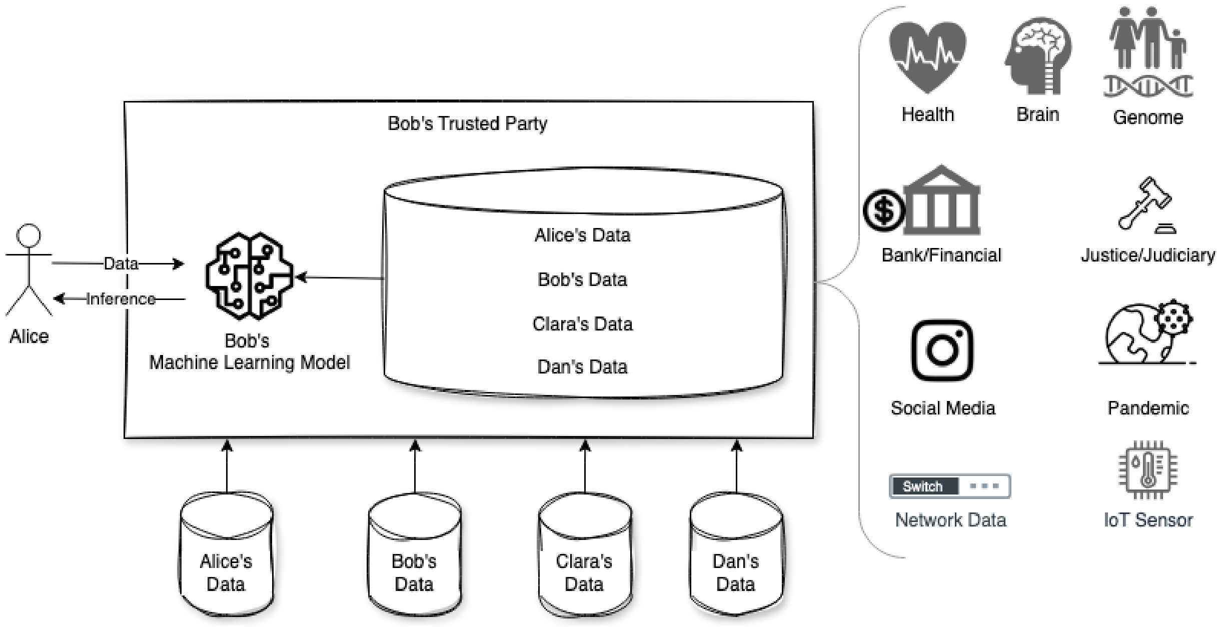

Figure 1.1: Example of applications that may require privacy requirements

may include customer information, revenue, profits, and salaries. While it is possible to remove sensitive data for legal reasons in some situations, this may not be feasible in others, as this information may be essential to improve the model's performance during training or needed in the inference phase. Therefore, appropriate methods are necessary for handling private information when the data used by ML algorithms are sensitive. The field of research that seeks to reconcile ML with privacy is known as PPML. This area is fundamental since sensitive data are essential for many ML models, especially in health [5], genetics, finance [9], and cybersecurity [3].

The motivation to explore this problem arises from social discussions about privacy [14]. An information-driven society collects vast amounts of sensitive data from individuals and companies. This problem is significant because data, when properly used, can generate numerous beneficial applications for individuals. For example, social networks use personal information to make personalized recommendations for news, books, clothes, travel, hotels, restaurants, movies, and music [11–13].

Therefore, ensuring data privacy in ML applications in healthcare, financial services, or cybersecurity is crucial, both for legal reasons and for the challenge of implementing Privacy-Enhancing Technologies (PET), which significantly increases the computational cost and execution time of ML-based solutions. The ideal privacy would be one in which the user's data is known only to the user; cryptographic solutions that require data decryption before being processed by ML algorithms would not meet this requirement.

This thesis proposes and implements a framework that ensures data privacy in ML model algorithms [15], with case studies focused on the detection of Malicious Software (Malware). The proposed framework preserves data privacy in the training and inference phases of model secure computing. P^{2}MLF inference uses Secure Multi-Party Computation (MPC) protocols and models trained with Differential Privacy (DP). P^{2}MLF proposes three three methods to train: the first combines Federated Learning (FL) with DP, the second combines MPC with DP, and the third involves FL with DP and aggregation with MPC, thus ensuring data privacy.

1.1 Problem

Privacy issues arise at both the input and output stages in ML applications. To illustrate these issues, consider that *Alice* owns the data and *Bob* owns the model. At the input stage, it is possible to expose the data needed for detection, compromising both the privacy of *Alice* 's data and the intellectual property of *Bob* 's model. Ensuring input privacy means that *Bob* does not need to reveal his model to *Alice*, and *Alice* does not need to expose her data to *Bob* 's model.

Similarly, classification results may reveal information about the model at the output stage, allowing attackers to exploit it and reverse engineer the training data [21]. Hence, the challenge is to enhance privacy in both the input and output stages during inference. It is important to note that training plays a crucial role in ensuring end-to-end privacy protection, adding a layer of complexity to the task.

Thus, the central problem to be solved is to propose a comprehensive approach that functions as a framework to ensure the privacy of input and output in ML applications.

1.2 Motivation for the problem

The right to personal data privacy is fundamental for individuals [10]. At the same time, the benefits provided by the ML applications are crucial, making it essential to find solutions that reconcile ML and privacy.

One of the main dilemmas to resolve is to perform ML training or inference without relying on a trusted third party. In other words, it is necessary to overcome the dilemma of using private data without allowing each party to access the confidential data of the different parties involved in the training and inference phases of the ML models.

Although private data could be stored securely and never used, this approach could be more practical, especially considering that such data can significantly improve the accuracy of ML algorithms.

The challenge lies in controlling the leakage of private data, considering a scenario where data is stored, communicated, or processed, even without requiring the data owner to rely on a trusted party for data transmission. This problem becomes particularly intriguing because private data from different sources can drive the creation of many valuable ML-based services. For example, researchers or organizations can combine sensitive information from multiple individuals to detect pandemics or diseases. Another example would be the detection of crimes related to child pornography while ensuring the privacy of suspect data is protected [17].

The figure 1.1 will be used to motivate the importance of ensuring the privacy of input and output in ML. In the inference phase in ML, input privacy may be necessary to protect *Alice*'s data and *Bob*'s model. In training, input privacy enables the creation of a global ML model that ensures the privacy of data from different parties, such as *Alice*, *Bob*, *Clara*, and *Dan*. Output privacy helps prevent *Alice* in the inference phase from obtaining information about the training data of *Bob*'s model.

1.3 Objectives

This thesis presents the P^{2}MLF framework, designed to ensure both input and output privacy in ML models while simultaneously optimizing efficiency in the use of MPC, FL, and DP technologies. The main objective is to reduce execution time and the number of network communication rounds while maintaining the accuracy of the trained models at levels comparable to those developed without privacy mechanisms.

In addition to the main objective, this work seeks to achieve the following secondary goals:

- Apply the P^{2}MLF framework to privacy-preserving Malware inference, using MPC protocols for MultiLayer Perceptron (MLP), One-Dimensional Convolutional Neural Network (CNN1D), and Long Short-Term Memory (LSTM) models. This work aims to identify the ML model that offers the best balance between privacy, lower execution time, and lower error rates among the three implemented MPC protocols.
- Demonstrate the application of P^{2}MLF to collaborative training, using a collaboratively trained intrusion detection system as a case study.

1.4 Results and Scientific Contributions

This thesis proposes an approach called P^{2}MLF to ensure data privacy in the training, pre-processing, and inference phases of ML and Deep Learning (DL) models. The application of P^{2}MLF will be demonstrated in the inference phase through the Domain Generation Algorithms (DGA) detection problem. In contrast, the approach will create a Collaborative Intrusion Detection Systems (CIDS) model using network data from various sources during the training phase. The methodology adopted in this thesis formalizes P^{2}MLF in structured blocks, integrated into a ML pipeline, allowing it to serve as a practical guide for professionals who need to incorporate privacy into ML applications. For these reasons, P^{2}MLF is presented as one of the main contributions of this thesis.

Some of the contributions of P^{2}MLF-based inference are listed below:

- The closest work on privacy-preserving detection of DGA is that of Drichel et al. [18], which provides only input privacy via MPC, leaving the inference phase vulnerable to privacy attacks on training data. This thesis fills the exit privacy gap identified in Drichel's work. Therefore, this thesis proposes the first end-to-end privacy-preserving framework for DGA classification, ensuring both input and output privacy.
- A method for *secure inference* from DGA using ML, with input privacy and output privacy guarantees.
- Original approach for DGA classification using ML with DP.
- The framework uses different MPC protocols such as MLP, CNN1D, and LSTM which are all combined into a MPC protocol for Embedding Layer (EL). The goal was to achieve a model with higher accuracy and shorter runtime.
- The MPC protocol for LSTM is the first to be built on MP-SPDZ [19].
- 23% to 42% reduction in runtime using post-training quantization with 16 bits, without reducing accuracy in a semi-honest MPC protocol.

Some of the contributions of P^{2}MLF-based training are listed below:

- Three methods for *secure collaborative training* from CIDS using ML, with guarantee of input privacy and output privacy.
- Comparison of the three collaborative trains concerning scalability and privacy.
- The method to a collaborative training of P^{2}MLF using aggregation with MPC is considered the better balance considering scalability and privacy, being 1.50 times faster than the approach with the highest level of privacy.

1.5 Publications and Competitions

The following is a list of published works related to this thesis's work.

- **Ricardo Jose Menezes Maia, Dustin Ray, Sikha Pentyala, Rafael Dowsley, Martine De Cock, Anderson C. A. Nascimento and Ricardo Jacobi. An End-to-End Framework for Private DGA Detection as a Service**. Plos One

Journal 2024 has accepted this paper for publication. The results of this paper support this thesis's work on the method for secure inference.

This paper results from the U.K.-U.S. prize challenges **Jelle Vos, Sikha Pentyala, Steven Golob, Ricardo Maia, Dean Kelley, Zekeriya Erkin, Martine De Cock, and Anderson Nascimento. Privacy-Preserving Membership Queries for Federated Anomaly Detection. 24th Privacy Enhancing Technologies Symposium (PETS 2024) 15-20 July 2024 Bristol, UK.** [1] [9]. My contribution to this paper was experiments with neural networks and DP. This paper is about a proposal to ensure data privacy and detect financial crimes using ML.

- **David Melanson; Ricardo Maia; Hee-Seok Kim; Martine De Cock; Anderson Nascimento. Secure Multi-Party Computation for Personalized Human Activity Recognition. 2022. Neural Processing Letters – Springer Journals.** [2]. My contribution to the paper is the MPC protocol to Convolutional Neural Network (CNN), which I also used in the DGA paper. This paper is about calibrating Human Activity Recognition (HAR) models to end-users with Transfer Learning (TL), which often yields significant accuracy improvements. By design, this type of TL relies on sensors worn close to the human body to collect personal data. To protect users' privacy, we introduce MPC protocols for personalizing HAR models and securing activity recognition with personalized models [22].

Throughout this thesis's research, I had the privilege and opportunity to participate in three competitions related to PPML, which is the area of this thesis. The competitions are listed below by impact factor:

- **U.K.-U.S. Prize Challenges: At the Summit for Democracy, the United Kingdom and the United States announced the winners of the Challenge to Drive Innovation in Privacy-Enhanced Technologies that Reinforce Democratic Values.** The US and the UK governments organized this challenge. I competed on the US team, named PPMLHuskies [3]. Our team consisted of the University of Washington Tacoma in the US, colleagues from the University of Delft in the Netherlands, and myself, representing the University of Brasília. We secured second place in this prestigious competition that involved American companies and universities. The competition aimed to detect financial crimes between banks in different countries while ensuring data privacy. My contribution to this competition consisted of conducting experiments and hyperparameter optimizations for the

[1] `https://petsymposium.org/popets/2024/popets-2024-0074.pdf`
[2] `https://link.springer.com/article/10.1007/s11063-023-11182-8`
[3] `https://petsprizechallenges.com/`

models trained with DP and participating in discussions until we arrived at the proposed solution.

- **Solution of Team PPMLRobots for PRIVACY SECURITY WORKSHOP - Secure Genome Analysis Competition (IDASH) 2021, Track III: A Differentially Private MPC Protocol for Logistic Regression**. I competed on the US team, PPMLRobots, and this collaborative work with colleagues from the University of Washington Tacoma in the US, Monash University in Australia, and me, representing the University of Brasília, submitted it to IDASH. In this global competition, universities compete around the world. We achieved first place [4], with the winner determined by the solution with the shortest execution time and highest accuracy. My contribution to this competition was the creation of MPC protocols for training logistic regression models using DP with Laplacian noise. The paper that consolidates the results of the competition IDASH is **Sikha Pentyala; Davis Railsback; Ricardo Maia; Rafael Dowsley; David Melanson; Anderson Nascimento; Martine De Cock**. Training Differentially Private Models with Secure Multiparty Computation. Computer Science -> Cryptography and Security. 2022. [5].

1.6 Outline of the Thesis

This thesis organizes its content into chapters that systematically address the research objectives. Chapter 1 presents the introduction to the work, including the problem description, motivation, contributions, published papers, and awards obtained in international competitions related to the thesis. Chapter 2 discusses the privacy techniques and theoretical foundations of the MPC, DP, and this work will address ML algorithms. Chapter 3 deals with works related to privacy-preserving DGA detection.

Chapter 4 describes the general framework proposed in this thesis, which is general-purpose but will be applied explicitly to the detection of DGA and Intrusion Detection System (IDS). The 5 chapter demonstrates methods for securely training a ML model using a combination of techniques such as FL, MPC, and DP and applies these methods to a IDS. Chapter 6 presents an application developed to detect Malware in inputs and outputs of private domain names, using the DP and MPC protocols with LSTM, CNN1D and MLP models, in addition to analyzing the results of the privacy-preserving DGA detection. Finally, Chapter 7 contains the conclusions and developments of future work.

[4] `http://www.humangenomeprivacy.org/2021/`

Chapter 2

Background

This chapter establishes the essential theoretical foundations necessary to understand and support the protocols proposed in this thesis.

The symbols x and y (x and y are scalars; $\mathbf{x}$ and $\mathbf{y}$ are vectors; $\mathbf{X}$ and $\mathbf{Y}$ are matrices) may have different meanings in subsequent sections depending on the context. The meanings will be explicitly defined whenever mentioned throughout the text to avoid ambiguity.

2.1 Privacy-Enhancing Technologies

This section discusses the main technologies used in this thesis to enhance data privacy, including Secure Multi-Party Computation (MPC), Differential Privacy (DP), and Federated Learning (FL).

2.1.1 Secure Multi-Party Computation

MPC protocols allow mutually distrustful parties to perform joint computations, ensuring that all honest parties obtain the correct result. No group of corrupt parties can obtain information beyond what is deducible from their inputs and outputs. In this work, we use a variant of traditional MPC, where computation servers offer MPC as a service, receiving inputs from external parties that do not participate directly in the protocol [15, 23–25].

The central principle of all MPC protocols is to decompose the function and to compute it into a circuit composed of addition and multiplication gates. The underlying MPC protocol executes and evaluates these gates sequentially until the final result is computed and revealed to the recipient [23].

Splitting functions into addition and multiplication circuits efficiently is not trivial. Optimized representations can significantly improve the performance of a MPC computation for a specific function.

The MPC protocols in this thesis use secret sharing, a technique in which a secret s is split into multiple shares x_i, distributed among shareholders. Only authorized combinations of shareholders can reconstruct the secret, and no individual shareholder can learn anything about the secret s.

MPC protocols based on secret sharing work by sharing the inputs among all computation servers. Computations occur in shares, not in the original inputs [15].

MPC involves n parties, $\{P_1, P_2, ..., P_n\}$, each with a private input x_i. The goal is to compute a function $y = f(x_1, x_2, ..., x_n)$, such that:

- Each party P_i learns only y, the output of f.
- No party learns information about the inputs of other parties x_j, where $j \neq i$.

This process uses an MPC protocol, which involves local computations and message exchanges based on the inputs x_i and random bits r_i sampled by each party. The protocol ensures that $y \leftarrow f(x_1, x_2, ..., x_n)$, while preserving the privacy of each entry x_i.

2.1.1.1 History of MPC

Andre Yao introduced the concept of secure computation in 1982 [26]. In this paper, Yao formulated the "millionaire problem" to illustrate his idea: "Two millionaires want to know who is richer, but without revealing any additional information about their wealth".

Later, in 1986, Yao presented new protocols for MPC, known as *Yao-garbling*, in [27]. The *garbling* technique showed that two participants can evaluate any function with computational security.

In 1987, Goldreich, Micali, and Wigderson [28] expanded this result to any number of participants as long as there was an honest majority based on the existence of one-way permutations with a trapdoor. This result emerged independently, followed by the work of Chaum, Damgaard, and van de Graaf [29]. It is worth mentioning that Yao's technique results in protocols with a constant number of rounds. A constant number of rounds for an arbitrary number of participants was achieved in 1990 by Beaver, Micali, and Rogaway [30].

In 1988, Ben-Or, Goldwasser, and Wigderson [31], independently of Chaum, Crepeau, and Damgaard [32], showed that in the secure point-to-point channel model, any function is possible to evaluate with unconditional security against up to T corrupted parties, where $T < \frac{n}{2}$ for passive adversaries and $T < \frac{n}{3}$ for active adversaries. In 1989, Ben-Or and Rabin [33] showed that achieving $T < \frac{n}{2}$ even against active adversaries is possible if broadcast and a small error probability is allowed.

2.1.1.2 Applications of MPC

Some possible applications for MPC include secure auctions, secret voting, secure Machine Learning (ML), and private database querying [15, 23–25, 34].

Consider, for example, the problem of comparing a person's DNA with a database of DNA from cancer patients to determine whether that person is in a high-risk group for a specific category of cancer. This task is critical to health and society, but DNA information is highly confidential, and one should not reveal it to private organizations. One can resolve this dilemma using MPC, which reveals only the cancer category that matches the person's DNA (or none). In this scenario, privacy is ensured by limiting disclosure to only the cancer category, without revealing additional information about the person's DNA or the patients in the database [35]. Furthermore, correctness ensures that no malicious party can alter the outcome, for example, by making the person believe they are unduly at risk.

Another example involves a trading platform, where parties submit bids and offers and then compare them to determine a match when the offer is higher than the bid. The trading price may be a function of the bid and offer. In this scenario, it may be beneficial from a game theory perspective to keep the actual values of the bids and offers private since other participants could use this information to manipulate prices. Privacy limits disclosure to only the match and the resulting price, while correctness verifies that the price aligns with the established function [23]. In some cases, privacy is more critical (as in the DNA example), while in others, correctness is paramount (as in negotiation). In any case, MPC guarantees both properties.

MPC protocols allow mutually distrusting parties to perform a computation so that all the honest parties receive the correct result at the end of the protocol. Colluding dishonest parties cannot gain any information about the inputs of honest parties beyond what their inputs and outputs allow them to infer. In this work, we will use a variant of the traditional MPC scenario, where the computation servers offer MPC as a service, with external parties providing the inputs that do not directly participate in the MPC protocol. For a more detailed introduction to MPC, we suggest reading works such as [15, 23–25].

The central idea of all MPC protocols is to decompose the function to compute into a circuit composed of addition and multiplication gates. The underlying protocol MPC is then used to evaluate each addition and multiplication gate sequentially until the result is computed and revealed to the designated receiver [36].

Decomposing functions into addition and multiplication circuits efficiently is a non-trivial task. Optimized representations can significantly improve the performance of a MPC computation for a specific function. We implement our solutions using the publicly available MP-SPDZ framework [19], which implements several protocols MPC. MP-SPDZ provides a high-level Python interface to present a circuit to compute in an MPC protocol. In addition, MP-SPDZ already has circuit representations for ML algorithms, including MultiLayer Perceptron (MLP) and 2D Convolutional Neural Network (CNN). However, to date, no implementations Long Short-Term Memory (LSTM) or Embedding Layer (EL) are available in the literature for MP-SPDZ. This work presents new circuit representations for LSTM and EL network inference, which have been implemented in MP-SPDZ [19].

In this work, we use secret sharing-based protocols MPC. Secret sharing consists of dividing an input s into multiple fragments (secret shares) x_i and distributing these fragments among different parties (shareholders). Only authorized combinations of these shareholders can reconstruct the secret, and no individual shareholder can obtain information about the value of s. Secret-sharing-based MPC protocols work by sharing the inputs secretly among all computing servers, and computations are performed directly on these shares instead of on the original inputs [15].

2.1.1.3 Basic primitives

This section will discuss basic primitives used to create MPC protocols.

2.1.1.4 Secret Sharing

Secret sharing is a fundamental primitive at the core of many approaches to MPC. Informally, a secret sharing scheme (t, n) divides the secret s into n *shares* such that any $t - 1$ of the *shares* reveal no information about the secret s. In contrast, any set of t *shares* allows the complete reconstruction of s. Several variants exploit different security properties of secret sharing schemes [36].

Let D be the domain of secrets and D_1 be the domain of *shares.* We define $Shr : D \to D_1^n$ as a (possibly random) sharing algorithm and $Rec : D_1^k \to D$ as a reconstruction algorithm. A secret sharing scheme (t, n) is a pair of algorithms (Shr, Rec) that satisfy the following properties:

- Correction: Let $(s_1, s_2, \ldots, s_n) = Shr(s)$. Then:

$$Pr[\forall k \geq t, Rec(s_{i_1}, \ldots, s_{i_k}) = s] = 1$$

- Privacy: Any set of shares with less than t *shares* does not reveal anything about the secret in the information-theoretic sense. Formally, for any two secrets $a, b \in D$ and any possible vector of *shares* $v = (v_1, v_2, \ldots, v_k)$ such that $k < t$:

$$Pr[v = Shr(a)|_k] = Pr[v = Shr(b)|_k],$$

where $|_k$ denotes the proper projection onto a subspace of k elements.

Consider the function f will calculate the values of $[\![x]\!]$ (known only to *Alice*) and $[\![x]\!]$ (known only to *Bob*). The value of $[\![x]\!]$ is split in secret shares (x_0, x_1), while $[\![y]\!]$ is split in secret shares (y_0, y_1). In the *secret sharing* paradigm, only *shares* are exchanged between the parties. For ease of understanding, only the *shares* x_0 and x_1, as well as the values y_0 and y_1, would remain private to the respective parties [36].

Alice and Bob compute on integers modulo an integer q, a hyperparameter defining their operations' algebraic structure. To secret-share a value x in $\mathbb{Z}_q \leftarrow \{0, 1, \ldots, q-1\}$, they select uniformly random values $x_0, x_1 \in \mathbb{Z}_q$ such that $x_0 + x_1 \equiv x \pmod{q}$. Alice and Bob hold x_0 and x_1 as additive shares of x. While neither x_0 nor x_1 individually reveal any information about the secret value x, they can easily reconstruct x by combining both shares. This approach enables Alice and Bob to perform computations on their respective shares collaboratively without exposing the actual values of the numbers [36].

2.1.1.5 Protocol for addition

Consider that *Alice* and *Bob* want to compute the function $[\![z]\!] \leftarrow f(x, y)$, where f is defined as $[\![z]\!] \leftarrow [\![x]\!] + [\![y]\!]$, $[\![x]\!] \leftarrow (x_0, \cdots, x_1)$, and $[\![y]\!] \leftarrow (y_0, \cdots, y_1)$. Therefore *Alice* and *Bob* computes $x + y$ using their local shares as follow $x + y \leftarrow x_0 + x_1 + y_0 + y_1$. In this work, π_{SUM} represents this protocol.

2.1.1.6 Protocol for addition by a constant

Consider that *Alice* and *Bob* want to compute the function $[\![z]\!] \leftarrow f(x, c)$, where f is defined as $[\![z]\!] \leftarrow [\![x]\!] + c$, $[\![x]\!] \leftarrow (x_0, \cdots, x_1)$, and publicly known constant $c \in \mathbb{Z}_q$. Therefore, *Alice* and *Bob* compute $x + c$ using their local shares as follows $x + c \leftarrow x_0 + c + x_1$.

2.1.1.7 Protocol for multiplication by a scalar

Consider that *Alice* and *Bob* want to compute the function $[\![z]\!] \leftarrow f(x, c)$, where f is defined as $[\![z]\!] \leftarrow c\,[\![x]\!]$, $[\![x]\!] \leftarrow (x_0, \cdots, x_1)$, and publicly known scalar $c \in \mathbb{Z}_q$. Therefore, *Alice* and *Bob* compute cx using their local shares as follows: $cx \leftarrow cx_0 + cx_1$.

2.1.1.8 Protocol for Multiplication

In this protocol for multiplication during the online phase, the parties generate multiplication triples (Beaver triples [37]) in the offline phase. Each triple, $([\![a]\!], [\![b]\!], [\![c]\!])$, consists of random values $a \leftarrow (a_0, \cdots a_n), b \leftarrow (b_0, \cdots b_n) \in \mathcal{Z}_q$, and $[\![c]\!] \leftarrow [\![a]\!] \cdot [\![b]\!]$, unknown to either party. These tuples are independent of the inputs and optimize the online phase by offloading expensive preprocessing work to the offline phase [36].

Alice and *Bob* want to compute the product of two shared values, $[\![x]\!]$ and $[\![y]\!]$, without revealing their complete values. They use a precomputed multiplication triple $([\![a]\!], [\![b]\!], [\![c]\!])$. *Alice* and *Bob* start by locally computing $[\![d]\!] \leftarrow [\![x]\!] - [\![a]\!]$ and $[\![e]\!] \leftarrow [\![y]\!] - [\![b]\!]$, then exchange their shares of $[\![d]\!]$ and $[\![e]\!]$ to reconstruct the public values d and e. Using d, e, and the triple, they compute $[\![z]\!] \leftarrow d \cdot [\![b]\!] + e \cdot [\![a]\!] + [\![c]\!] + d \cdot e$, obtaining a shared value of $z = x \cdot y$ without revealing the individual values of x and y.

Replacing the terms of $d \leftarrow x - a$ and $e \leftarrow y - b$ into the expression $[\![z]\!] \leftarrow d \cdot [\![b]\!] + e \cdot [\![a]\!] + [\![c]\!] + d \cdot e$, and considering that $[\![c]\!] \leftarrow [\![a]\!] \cdot [\![b]\!]$, the expression simplifies to $z \leftarrow xy$, ensuring that the product of the original shared values is calculated correctly.

2.1.1.9 Adversarial Models

A more formal definition helps clarify the security properties that MPC offers. The real-world and ideal-world paradigms form the conceptual basis for defining security.

2.1.1.10 Real-World and Ideal-World Paradigms

One way to define security is to list the conditions that constitute a security violation. For example, the adversary must not be able to learn a specific predicate from another party's input, induce incorrect outputs for honest parties, or make its inputs depend on the inputs of honest parties. However, this approach is cumbersome and error-prone since it is not trivial to ensure that the list of conditions is complete [23].

To provide a mathematical proof of security, it is necessary first to define what it means for a protocol to be secure. This proof is difficult to formalize in the context of MPC since the parties need to learn the output, which depends on the inputs, and one cannot simply say that the parties "learn nothing". Furthermore, the "correct" output depends on the inputs, and it is not known in advance which inputs the corrupted parties will use [23].

The real/ideal paradigm solves this difficulty by introducing an "ideal world" that implicitly captures all security guarantees and defines security relative to that world [23].

Ideal World

In the ideal world, the parties securely compute the function F by sending its inputs privately to a fully trusted party T. Each party P_i provides its input x_i to T, which computes $F(x_1, ..., x_n)$ and returns the result to all parties [23].

One can imagine an adversary attacking this interaction in the ideal world. The adversary can take control of any party P_i, but not T (which justifies trust in T). The simplicity of the ideal world makes it easy to understand the effects of an attack [23].

Although the ideal world is easy to understand, the presence of a trusted third party makes it imaginary. The ideal world serves as a benchmark for judging the security of a protocol in the real world [23].

Real World

In the real world, there is no trusted party. Instead, all parties communicate using a protocol [23].

In the real world, an adversary can corrupt some parties, and corruption at the beginning of the protocol is equivalent to the original party becoming an adversary. Depending on the threat model, corrupted parties may follow the protocol as specified or arbitrarily deviate from its behavior. Intuitively, the real-world protocol π is secure if a corresponding adversary in the ideal world can achieve any effect that the real-world adversary could achieve. In other words, the goal of a protocol is to provide security in the real world (given a set of assumptions) equivalent to security in the ideal world [23].

2.1.1.11 Semi-honest or curious

A semi-honest, or curious, adversary corrupts the parties but follows the specified protocol. This type of adversary is also called passive or "honest but curious". The corrupted parties execute the protocol correctly but may attempt to extract as much information as possible from the messages they receive from other parties. This process may involve multiple corrupt parties colluding to obtain information. Semi-honest adversaries are passive because they cannot take any action other than to attempt to obtain private data by observing the execution of the protocol [23].

In the real/ideal paradigm, security implies that such an "attack" could also be performed in the ideal world. For a protocol to be considered secure in the ideal world, generating something indistinguishable from the adversary's view of the real world must be possible. The adversary's view in the ideal world consists only of the inputs sent to the trusted third party T and the outputs received from T. Thus, an ideal-world adversary should be able to use this information to generate what appears to be a view of the real world. Referring to an ideal-world adversary as a simulator is possible because it generates a "simulated" view of the real world while operating in the ideal world. Showing that such a simulator can exist demonstrates that an adversary can do nothing in the real world that cannot be done in the ideal world [23].

It is possible to formalize as follows: Let π be a protocol and F be a functionality. Let C be the set of corrupted parts and Sim be the simulator. The following distributions of random variables are defined:

- $\text{Real}_\pi(k, C; x_1, ..., x_n)$: execute protocol with security parameter k, where each party P_i executes protocol honestly with private input x_i. Let V_i denote the final view of the P_i part, and let y_i denote the final output of the P_i part. Output $\{\ V_i | i \in C\ \}$, $(y_1, \ldots, y_n)$.
- $\text{Ideal}_{F,Yes}(k, C; x_1, ..., x_n)$: Compute $(y_1, \ldots, y_n) \leftarrow F(x_1, \ldots, x_n)$.

  Output Yes $(C, \{(x_i, y_i) | i \in C\}), (y_1, \ldots, y_n)$.

A protocol is secure against semi-honest adversaries when corrupt parties in the real world hold views indistinguishable from those they would have in the ideal world.

2.1.1.12 Malicious

A malicious adversary, also known as an active adversary, can cause corrupt parties to deviate from the prescribed protocol to compromise security arbitrarily. A malicious adversary has all the powers of a semi-honest adversary to analyze the execution of the protocol but can also perform any actions during execution. The actions of this malicious adversary may include controlling, manipulating, and injecting messages into the network. Protocols that ensure security in this model offer very high protection. In the case of a dishonest majority, the most an adversary can do is force the honest parties to abort the protocol upon detecting cheating. If the honest parties obtain results, these are guaranteed to be correct and always preserve privacy [23].

2.1.1.13 The need for a trusted third party

The problem arises when trying to find a third party, T, that is sufficiently trustworthy for all parties to share their secrets. In this scenario, the parties give their inputs to T, which performs the computations, announces the results to each party, and then deletes the confidential data to which it has access. However, this approach presents security problems: (i) T is a single point of attack, making it vulnerable to private data extraction, and (ii) the parties need to trust T to ensure both the accuracy of the results and the confidentiality of the data. The question arises: Why would the parties trust T if they do not trust each other [25]? Although a trusted third party would work in an ideal scenario, it does not exist in the real world.

This question arises: solving the problem without a trusted third party when one wants to compute a result that depends on private data from all parties involved [25].

2.1.1.14 Outsourced Computation

Secure computation encompasses methods that perform computations on sensitive data. In contrast, verifiable computation allows participants to confirm that the result is indeed the correct output of the function on the given inputs. Two main categories are outsourced computation and multi-party computation [23].

In an outsourced computation, one party owns the data and wants to obtain the result of a computation. The second party receives the data in an encrypted format, performs the computations on the encrypted data, and returns the encrypted results to the data owner without learning anything about the input information, the intermediate values, or the result [23].

The data owner decrypts the results to obtain the output. Homomorphic Encryption (HE) allows for operations on encrypted data and is a suitable primitive to implement third-party computation [23].

2.1.2 Universal Composability

Universal Composability (UC) serves as a framework for cryptographic protocol design, ensuring that security properties remain intact when protocols are composed or executed concurrently [38]. Consider a protocol Π_F that securely implements a functionality F in the presence of an auxiliary functionality R, a hybrid model. While R acts as a trusted intermediary in this model, it must be instantiated in practice, typically by another protocol Π_R, which might rely on a simpler functionality T. The UC framework's core result is that the composed protocol inherits the security properties of the involved protocols: Π_F securely implements F, even when replaces R by Π_R in the T-hybrid model. This modularity enables the breakdown of complex protocols into simpler components, with security proofs combining each part to ensure the overall system's security. Additionally, the UC framework supports concurrent execution of multiple protocols in distributed systems, maintaining security even when different protocols interact. Making UC mighty compared to stand-alone models facilitates robust, modular, and scalable protocol design and analysis [23, 36].

2.1.3 Differential Privacy

Informally, a differentially private algorithm produces a given output with approximately the same probability, regardless of whether a single entry is present or absent in a dataset used to compute the algorithm output. Meaning that the output is negligibly affected by the participation of a single user, offering privacy through plausible deniability [39].

This work presents the definition of DP as described in [39]:

Definition 1. *(ϵ, δ) - DP*

A randomized algorithm $\mathcal{M}$ with domain $\mathbb{D}$ is (ϵ, δ) - differentially private if for all $S \subseteq$ Range($\mathcal{M}$) and for all $x, y \in \mathbb{D}$ such that $||x - y||_1 \leq 1$:

$$Pr[\mathcal{M}(x) \in S] \leq exp(\epsilon) Pr[\mathcal{M}(y) \in S] + \delta \tag{2.1}$$

The definition 1 implies that queries in datasets x and y that differ in a single input ($||x - y||_1 \leq 1$) should produce different results with a bounded probability, depending on the parameters ϵ and δ. The constant ϵ is the *privacy budget*. The smaller the value of ϵ, the more privacy the randomized algorithm $\mathcal{M}$ provides. The constant δ captures a small probability of violating the privacy guarantee. When δ equals zero, the algorithm is considered pure DP. When $\delta > 0$, the algorithm is said to be an approximate DP. Generally, choosing δ heuristically and empirically to be less than or equal to the reciprocal of the dataset size [39].

2.1.3.1 Differentially-Private Stochastic Gradient Descent

Deep learning models often leak information about their training datasets. This leakage can occur even with limited access to the model when an adversary only observes the output of the deep neural network. DP is used to prevent such leaks. It is possible in the training deep learning models with DP guarantees using Differentially-Private Stochastic Gradient Descent (DP-SGD) [40].

The two essential steps in DP-SGD, compared to traditional Stochastic Gradient Descent (SGD), are gradient clipping and noise addition. Gradient clipping limits the magnitude of gradients to a predetermined threshold, ensuring that the gradients computed for each data point decrease if they exceed this threshold. Preventing individual data points from disproportionately impacting the model's learning by reducing the model's sensitivity to any single piece of data. After clipping, the process adds noise to ensure privacy. The process typically samples this noise from a Gaussian distribution, setting the standard deviation or scale parameter proportional to the chosen privacy budget ϵ. Lower values of ϵ provide better privacy but require adding more significant amounts of noise, which can compromise the model's accuracy [40].

DP-SGD ensures $(\epsilon, \delta)-$DP through repeated applications of the Gaussian mechanism. During model training, *Bob* performs the following steps:

- Initialize the model M with random parameters.
- For each input x_i in a batch B, compute the model output $M(x_i)$ and, given the corresponding label y_i associated with x_i, compute the gradient of the loss function, denoted g_i.

- For each gradient g_i in batch B, clip the gradient if its L2 norm is greater than or equal to a threshold c.
- Average all clipped gradients in batch B and add, to each gradient coordinate, a random variable sampled from a Gaussian distribution with mean 0 and standard deviation $\sigma.c$, where c is the clipping threshold, and σ is a constant chosen to ensure DP.
- Update the model parameters using these aggregated, noisy gradients multiplied by the learning rate.
- Repeat this process for each batch in the dataset.

The post-processing property of DP ensures that any output derived from the model also maintains DP.

2.1.4 Federated Learning

In FL, a collaborative approach ML, users benefit from shared models trained on data from multiple sources without the need for centralized information storage. In this technique, a group of participating devices, called clients, collaborate to solve the learning task under the coordination of a central server [41].

Each client trains the model using its local data without sharing or transferring these data to the central server [42]. Instead, clients compute updates based on their local data to improve the global model maintained by the server. The clients send only these updates back to the central server. In this way, the global model benefits from the knowledge of all clients without exposing their raw data or leaving their devices. This approach preserves data privacy while enabling collaborative model improvement, ensuring customers contribute to ML securely and privately [41].

FL allows models to be trained locally without sharing private data but is vulnerable to backdoor attacks by compromised parties. In these situations, an adversary manipulates the global model by sending malicious updates that trigger unwanted behavior. Furthermore, FL does not prevent attacks such as membership inference, where one of the parties having access to the model can obtain information used by the other parties in training the local model.

2.2 Machine Learning and Deep Learning

Next, this work will address the ML and deep learning algorithms used in this thesis.

ML is an area of knowledge that applies statistical techniques through computational algorithms to identify patterns in data, with the aim of generalizing learning to new contexts. ML is divided into supervised learning, which uses labeled data, and unsupervised learning, which explores patterns in unlabeled data, distinguishing itself for using computers to estimate complex functions adjusted by hyperparameters. It is possible to define Deep Learning (DL) as a subarea of ML that uses deep neural networks to learn hierarchical representations of data using optimization algorithms. The main characteristic of DL is the ability to solve complex problems through structures composed of multiple processing layers [1].

2.2.1 Logistic Regression

Statistical prediction using binary Logistic Regression (LR) determines the probability that an observation falls into one of two categories based on a binary outcome variable. This prediction depends on one or more independent variables that can be continuous or categorical. The process involves constructing a logistic regression equation that calculates the probabilities' natural logarithm (LN), known as the logit transformation [43, 44].

$$\text{logit}(p) \leftarrow \ln\left(\frac{p}{1-p}\right) \leftarrow \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k \tag{2.2}$$

In equation LR 2.2, p represents the probability that an observation belongs to one of the two categories, while β_0 is the model constant. The undetermined parameters $\beta_0, \beta_1, \ldots, \beta_k$ must be estimated from the data and $x_0, x_1, \ldots, x_k$ represent the independent variables.

2.2.2 Multilayer Perceptron

An MLP is a ML model that represents a fully connected neural network architecture. The structure includes an input layer, an output layer, and one or more hidden layers, where neurons in adjacent layers are fully connected.

Equation 2.3 outlines the operation of a neuron in a MLP computing the output y and using an activation function σ. The function σ receives its argument by computing the dot product between the input vector of the neuron $\mathbf{x}$ and the weight matrix $\mathbf{W}$ and adding the vector bias $\mathbf{b}$.

$$y \leftarrow \sigma(\mathbf{x} \cdot \mathbf{W} + \mathbf{b}) \tag{2.3}$$

2.2.3 Long Short-Term Memory Networks

Understanding and reasoning about information in an abstract way depends on prior experience. For tasks such as reading text, humans extract knowledge based on the context of previous and recent pieces of content. Traditional neural networks do not have memory structures analogous to the human brain. Recurrent Neural Network (RNN) addresses this limitation, which more closely resembles memory processes. Thus, with a RNN, it is possible to perform the training and to learn based on previous elements of an input sequence.

RNNs are well suited to model problems where the input data is time-dependent or sequential, where the next task depends on the previous one. Examples include temperature/weather forecasting, historical air quality trends, and traffic congestion patterns. Furthermore, in Natural language processing (NLP), the meaning of texts and language structures depend on the content of previous texts.

LSTM is a type of RNN, is designed to solve the problem of vanishing and exploding gradients, common in traditional RNN models that deal with long-term dependencies [45]. LSTMs are particularly effective for Domain Generation Algorithms (DGA) detection [46–54]. Therefore, this work investigates the effectiveness of LSTM in the DGA detection problem, also considering the preservation of privacy.

The central concept of LSTM networks is the cell state, which functions as an internal memory. LSTM cells have gates that control the flow of information, allowing the network to add or remove information from the cell state, which acts as an adjustable memory system.

This work describes the functioning of a LSTM cell and its use for inference, i.e., to compute the outputs of a trained LSTM. The notation and explanation follow those presented in [55]. In the implementation, the input to the network is a sequence of characters $x_1, \ldots, x_t, \ldots, x_n$. Section 2.2.5.1 explains converting each character into a numeric representation. The inputs to the t-th cell consist of the output of the previous cell h_{t-1}, the character x_t, and the state of the previous cell c_{t-1}. The cell outputs h_t, and its state is c_t. The following section explains how to calculate these quantities.

First, we define how much of the previous state c_{t-1} will be "forgotten", denoted by the parameter f_t in Equation 2.4. The value of f_t is calculated based on the dot product of the weights w_f and the concatenation of the output of the previous cell h_{t-1} with the input of the cell x_t, plus the bias b_f. The training process learns with the weights and biases. This value is passed to the function *sigmoid* σ, which returns values between 0 and 1. When f_t equals 1, the process is to retain the entire previous state. In contrast, when f_t equals 0, the process fully discards the previous state while calculating the new cell state.

$$f_t \leftarrow \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.4}$$

Equation 2.7 shows the calculation of the cell state at position t. It is a weighted average between the previous state of the cell, c_{t-1}, and the state that depends on the current input, x_t, denoted as $c_t^{'}$. This average uses f_t and i_t as weights, with i_t determining the portion of the "current" state ($c_t^{'}$) to retain. The values of i_t and $c_t^{'}$ are obtained according to Equations 2.5 and 2.6, respectively. The parameters w_i and w_c represent the weights, while b_i and b_c are the biases adjusted during training. The function $tanh$ refers to the hyperbolic tangent.

$$i_t \leftarrow \sigma(w_i \cdot [h_{t-1}, x_t] + b_i]) \tag{2.5}$$

$$c_t^{'} \leftarrow tanh(w_c \cdot [h_{t-1}, x_t] + b_c]) \tag{2.6}$$

$$c_t \leftarrow f_t \cdot c_{t-1} \cdot i_t \cdot c_t^{'} \tag{2.7}$$

Finally, the cell output at position t, denoted as h_t, is calculated according to Equations 2.8 and 2.9. In these equations, w_o represents the weights and b_o the biases.

$$o_t \leftarrow \sigma(w_o \cdot [h_{t-1}, x_t] + b_o]) \tag{2.8}$$

$$h_t \leftarrow o_t \cdot tanh(c_t) \tag{2.9}$$

The variables h_t, c_t, and x_{t+1} pass to the next cell, where the calculations proceed similarly for the cell at position $t + 1$.

2.2.4 Convolutional Neural Network

A CNN typically consists of convolution blocks followed by a fully connected network. A typical convolution block includes a convolution layer, an activation layer, and a pooling layer. The standard convolution layer (2D-CNN) takes a three-dimensional input with height h, width $\mathbf{w}$, and depth c, and contains f 2D learnable kernels, each of size $k \times l$, applied to each input channel c. These kernels traverse in two directions over the input, generating a three-dimensional output.

In a One-Dimensional Convolutional Neural Network (CNN1D), the input is typically a sequence or vector, and the kernel traverses in only one direction [56].

Consider **x** as the input of a CNN1D, **y** as the output of the CNN1D layer, and k as the total number of kernels. The length of y is given by $l - k + 1$, assuming no padding is applied. The kernel performs a sliding window operation on the input **x**.

Equation 2.10 expresses the output of a CNN1D, where $\mathbf{y}[i]$ denotes the output at position i. The operation involving **x** and **w** is a dot product; b is the bias; **w** represents the trainable weights of the CNN1D including the kernels, and $\mathbf{w}[j]$ is the kernel at position j.

$$\mathbf{y}[i] \leftarrow \sum_{j=0}^{k-1}(\mathbf{x}[i+j] \cdot \mathbf{w}[j]) + b, \tag{2.10}$$

2.2.5 Preprocessing

In this section, the concepts of EL are used to improve the accuracy of detecting DGA.

2.2.5.1 Embedding layer

EL allows texts representing vectors of real numbers with finite precision. In this work, each vector represents a letter of the domain name. Instead of manually defining the embedding values, the method treats them as trainable parameters [3].

The process of obtaining the output of EL is described by Equation 2.11:

$$\mathbf{Y} \leftarrow \mathbf{X} \cdot \mathbf{W}, \tag{2.11}$$

The matrix **X**, dimension $l \times c$, contains one-hot encoded entries in the equation. In contrast, the output of EL, the matrix **Y** of dimension $l \times d$, represents embedding the input domain name to be classified. The matrix **W**, of dimension $c \times d$, is the embedding matrix, where l is the length of the input domain, c is the size of the character set, and d is the dimensionality of the vector space [3].

EL is used in this work to demonstrate secure inference, being the first layer in the DGA detection models. Removing the EL causes the accuracy of the DGA detection models to drop by more than 10%.

The use of EL in this context serves to more efficiently represent the characters of the Domain Name System (DNS) domains. The pre-processing, in this case, converts each character in a domain DNS into its ASCII value, and the matrix generated by EL maps these values into a weight vector, which serves as input for the subsequent layers of the model.

During training, EL is fine-tuned, learning to generate a weight matrix that optimally represents each character in a DNS domain, thus improving the performance of subsequent layers of the model.

2.2.6 Post-Processing

In this section, the concepts of quantization, which aims to improve the performance of MPC protocols applied to DGA detection, reducing execution time and communication overhead, will be presented.

2.2.6.1 Quantization

The precision used to represent the parameters of a ML model significantly impacts the model's accuracy, runtime, and size. This effect becomes even more pronounced when executing ML models with MPC protocols. Quantization, which reduces the accuracy of the models, improves the overall runtime efficiency and reduces communication in MPC protocols [3].

The detection of DGA uses quantization with Float16 post-training method[1], which converts the model weights from 32-bit floating-point numbers to 16-bit floating-point numbers, reducing the model size by half with minimal loss of accuracy. This quantization decreases the secure inference DGA runtime by approximately 23% to 42%.

2.3 Applications

This section presents two applications that exemplify the use of the framework proposed in this thesis: DGA and Intrusion Detection System (IDS).

2.3.1 Domain Generation Algorithms

Malicious Software (Malware) refers to a class of software that infects computers to perform unauthorized actions on the system or gain access to confidential information. This type of software is a significant source of illicit activity, causing increasingly negative impacts [57–60] and resulting in substantial losses in sectors such as government, energy, and manufacturing [61]. Examples of Malware families, such as Trojans, viruses, ransomware, keyloggers, worms, spyware, and hidden cryptominers, have as their primary goals information theft, espionage, and service disruption [3, 57, 58].

[1]`https://ai.google.dev/edge/litert/models/post_training_quantization`

Botnets, networks of computers infected with Malware, are typically controlled and updated through communication with a Command and Control Server (CCS) under the control of an adversary or botmaster [62]. When the IP address of the CCS server is hardcoded directly in Malware, IDS systems or firewalls on the DNS servers can block detected malicious domains, preventing connection to CCS and rendering Malware ineffective. However, sophisticated cyberattacks use innovative techniques to obfuscate the identity of the CCS, with the use of DGA being one of the most prevalent approaches [3,63].

DGA is an algorithm that periodically generates pseudo-random combinations of characters or words to create hundreds or thousands of new domain names, producing artificial malicious domains. The central idea is that a DGA can generate identical lists of new domains when executed on different machines, such as the botmaster and an infected host. The botmaster registers one or more generated domains, and the infected machines systematically query these domains until they successfully resolve one. If the botmaster does not register the queried domain, the infected machine receives a response of a non-existent domain and discards the attempt. Once the infected bot locates a registered domain, it establishes communication with the CCS, enabling the malware to execute the malicious activities directed by the CCS. DGAs play a crucial role in malware that relies on networked communication between the botmaster and bots (infected clients) [3, 63–65].

The malware's ability to dynamically generate new domain names prevents traditional blocklists from being effective in permanently blocking communication between the botmaster and infected machines [3]. If a firewall or IDS detects and blocks a domain, the botmaster can use DGA to generate and register a new domain, which the malware will use to establish a new connection. This work aims to perform the training and use neural networks capable of distinguishing between domains generated by DGA and benign domains, using ML models to recognize these domains in DNS traffic and mitigate the damage [3, 66].

The constant change of CCS domains makes it more difficult for IDS systems and firewalls to detect and contain attacks. Identifying generated malicious domains is the main challenge in mitigating DGA attacks. DNS servers need to be able to block these malicious domains while maintaining regular operation for benign domains. Identifying malicious domains can drastically reduce the damage caused by Malware [3].

As illustrated in Fig. 2.1, the central idea is that the botmaster and the malware in infected bots run the same DGA, with the same seed to generate identical lists of artificial domains. The botmaster registers one or more of these automatically generated domains, while Malware in infected bots attempts to resolve each domain through DNS.

As illustrated in Figure 2.1, the botmaster and the malware in the bots use the DGA "xxhex" family to generate a set of domain names, such as "xxd80cd4e0.cn" and

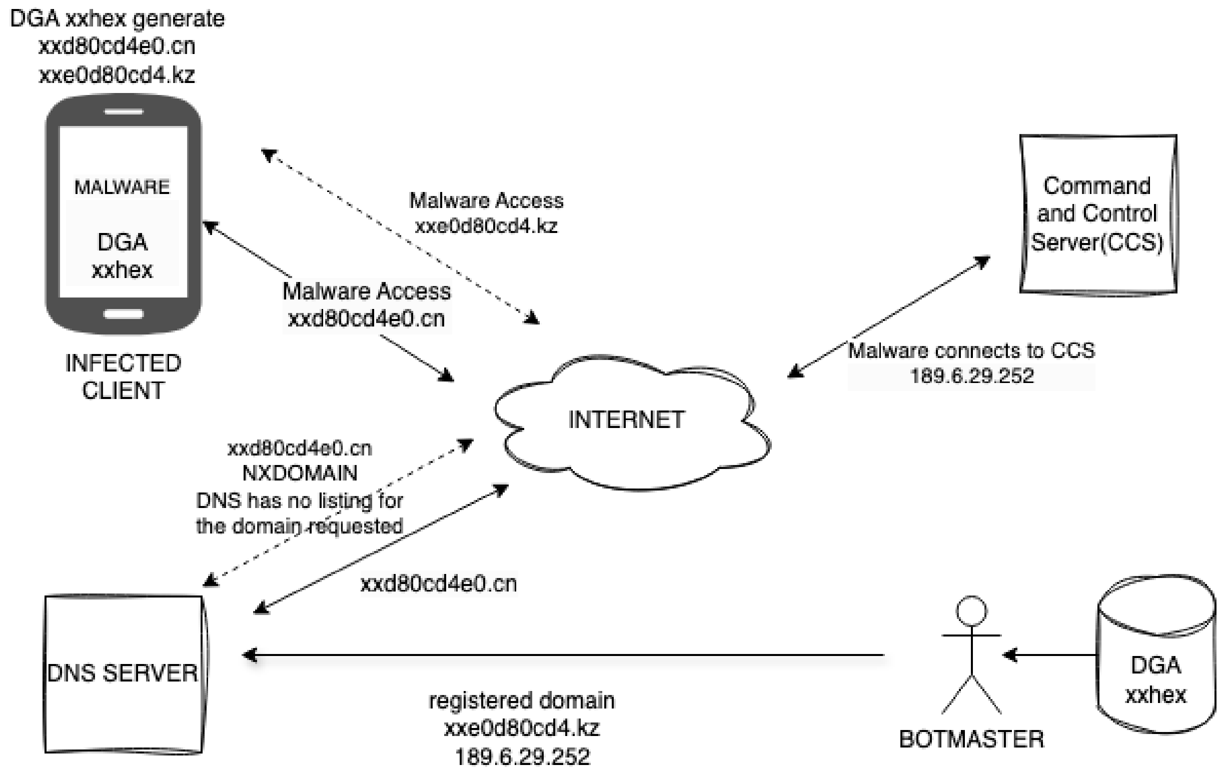

Figure 2.1: Illustration of DGA usage: The botmaster and malware on an infected client generate an identical list of domain names. The botmaster registers one of these generated domains. The malware then attempts to resolve each domain in the list via DNS queries until it finds the registered domain, thus establishing a successful connection between the infected client and the CCS server.

"xxe0d80cd4.kz", among others. The botmaster registers the domain "xxe0d80cd4.kz" and associates it with the IP address "189.6.29.252". When infecting a client, the malware attempts to resolve each generated domain through DNS queries. In Figure 2.1, when trying to resolve the domain "xxd80cd4e0.cn", DNS returns a non-existing domain (NXDOMAIN or NXD) error, indicating that the domain is not registered. On the other hand, when resolving the domain "xxe0d80cd4.kz", DNS returns the IP "189.6.29.252". The malware then connects to this IP address, where CCS is registered, thus establishing communication between the botmaster and the infected machine [3].

2.3.2 Intrusion Detection System

IDS protect networks by monitoring traffic to identify and categorize threats. However, these systems often monitor only a portion of the network, which can result in potential blind spots. For example, Distributed Denial of Service (DDoS) attacks, which pose major cybersecurity risks, can challenge the effectiveness of IDS when operating in isolation. Furthermore, different IDS have varying levels of effectiveness in detecting different types of attacks [67–69].

This proposal presents a Collaborative Intrusion Detection Systems (CIDS) that seeks to overcome these limitations. CIDS is trained with data from different sources, ensuring the privacy of each participant. The main advantage of this approach is that it maintains data privacy while identifying attack patterns that individual datasets might miss. Since companies deal with sensitive user information, privacy preservation is essential when defining specific solutions for CIDS.

The proposed framework allows stakeholders to collaborate to build a model that takes advantage of each dataset's unique strengths while prioritizing information privacy.

Chapter 3

Related Works

This work presents the related literature and compares the proposed solution and previous approaches.

The related work organizes the content according to two specific objectives of this thesis: Domain Generation Algorithms (DGA) and Intrusion Detection System (IDS) detection with privacy guarantees. Regarding DGA and IDS, the focus will be on studies that use Privacy-Enhancing Technologies (PET), such as Differential Privacy (DP), Secure Multi-Party Computation (MPC), and Federated Learning (FL). For DGA detection, a topic in Natural language processing (NLP) is included because the Domain Name System (DNS) domains are text, and the NLP techniques that generate embeddings of these domains have shown the best accuracy results.

3.1 DGA detection using deep learning

Initially, DGA detection methods did not utilize Machine Learning (ML). For example, Sharifnya et al. [70] developed a technique that identifies hosts with a high volume of failed DNS queries by adding these hosts to a "suspicious failure matrix". This section lists relevant work on DGA detection using deep learning or ML without privacy guarantees.

Li et al. [71] propose real-time detection models and frameworks that utilize domain-generated metadata, combining the advantages of a deep neural network and a lexical-resource-based model through ensemble. Another work along these lines is [72], which uses the Helix architecture, representing DGA as embeddings. The use of MultiLayer Perceptron (MLP) for the detection of DGA was also explored in [73] and [74].

Huang et al. [75] propose the Helios architecture, which uses Convolutional Neural Network (CNN) to detect DGA. Zhou et al. [76] also use One-Dimensional Convolutional Neural Network (CNN1D) for DGA detection, allowing binary and multiclass analysis.

Berman [77] employs CNN1D with a one-dimensional convolutional layer to detect DGA. Chen et al. [78] apply CNN1D and BiGRU to detect DGA.

Shahzad et al. [79] use Recurrent Neural Network (RNN) to detect DGA based only on the domain name without additional information, comparing the performance with other RNN architectures such as Unidirectional Long Short-Term Memory (LSTM), Bidirectional LSTM (Bi-LSTM) and Gated Recurrent Unit (GRU).

Zhang et al. [80] design and implement DGA classifiers using ML (SVM and RF) and deep learning (CNN, LSTM, and Bi-LSTM) methods. Yang et al. [81] explore character-level features of DGA domain names and propose a heterogeneous deep neural network framework including CNN1D and LSTM. The use of LSTM for binary and multiclass detection of DGA based on alphanumeric domain names was studied in [46, 47]. Tran et al. [49] present a novel LSTM algorithm to address the multiclass imbalance problem in DGA-related botnet detection. Strategies to deal with imbalanced datasets are essential for multiclass detection of DGA. Balakrishna et al. [82] utilize LSTM to improve the prediction of imbalanced datasets. Josan et al. [50] employ bidirectional LSTM for binary and multiclass detection of DGA. Other applications of LSTM in DGA detection are in [51, 52, 54, 67].

Liu et al. [83] combine CNN and bidirectional LSTM to detect DGA. Yun et al. [84] use natural language processing and Wasserstein Generative Adversarial Networks (GAN) to prevent attackers from evading DGA detection.

DGA detection in Internet of Things (IoT) scenarios was investigated by [85], focusing on its relevance for Malicious Software (Malware) detection in IoT devices.

Li et al. [86] use Hidden Markov Models (HMM) for inference. Koh et al. [87] use pre-trained context-aware word embeddings to classify DGA. Cucchiarelli et al. [88] use the Kullback-Leibner divergence and Jaccard Index to estimate similarities in DGA detection. Yilmaz et al. [89] apply LSTM and a GAN to identify unknown malicious domains.

Yu et al. [64] observe that simpler architectures tend to be faster in training and inference and are less prone to overfitting, a key finding for this work, which seeks lightweight architectures with better performance in MPC protocols. Yu et al. [66] propose heuristics to label domain names monitored in real traffic automatically, and [65] propose a method to label large volumes of real traffic, allowing training of models with real data.

Sivaguru et al. [90] strengthen DGA detectors against adversarial attacks by evaluating deep learning models and random forests (RFs) for DGA detection based on information beyond the domain name.

3.2 Secure Multi-Party Computation for DGA Detection

The closest work to ours is that of Drichel et al. [18], who also propose a framework and protocols for private inference and classification of DNS traffic. However, their runtimes and accuracy are not directly comparable to ours due to the weaker privacy guarantees they achieve. The work of Drichel et al. [18] allows the model information to leak *Bob* (such as embeddings). Furthermore, their solution fails to provide output privacy because it does not involve training the model with DP. Even with these weaker guarantees, their solution does not outperform ours in terms of runtime. Their models have inference times greater than 1 seconds, and their experiments do not consider network delays. The most accurate classifier in our solution (CNN1D) runs in less than 0.4 seconds, including network delays.

Drichel et al. [18] proposed a solution for MPC-based private inference applied to DGA detection models. The approach used several publicly available MPC frameworks for its implementation. At the time, there were no protocols or implementations available to perform word embedding (Embedding Layer (EL)) privately, leading the authors to assume that EL was publicly accessible and implemented by the party responsible for domain classification. However, this approach may leak information about the model, especially if EL was trained with private data [3].

In terms of communication complexity, the models presented in [18] exchange more than 190 MB in messages, while our fastest protocol exchanges 17 MB (with quantization) and the most accurate exchanges 21 MB (with quantization).

The work of Drichel et al. [18] uses MPC protocols to classify domains as DGA or non-DGA, implementing their proposals in different MPC frameworks, such as PySyft, TF-Encrypted, MP2ML and SecureQ8, applied to classifiers such as Inline [65], NYU [64], ResNet [62] and FANCI.

Despite being a pioneering approach, the work of Drichel et al. [18] has several limitations:

- It uses CNN-2D for the private inference of DNS domains, which adds unnecessary complexity, since CNN1D is more appropriate for text classification problems.
- It does not use privacy-preserving EL, which implies that *Bob* needs to leak the embeddings (EL) to *Alice* so that she can perform the embedding operation on the plaintext. Thus, the solution does not guarantee end-to-end privacy.
- It does not consider the use of LSTM models with MPC.
- It is limited to binary classification, while this work presents predictions for detecting binary and multiclass of DGA.

- Our work is the first to delve into the privacy-utility trade-off of Differentially-Private Stochastic Gradient Descent (DP-SGD) in the context of DGA classification. This unique perspective and our pioneering research make our work a compelling read for those interested in the intersection of privacy and utility in deep learning.

3.3 Secure Multi-Party Computation for Natural Language Processing

Hao et al. [91] present a solution for private inference in BERT transformer-based models in a client-server setting, where clients have private inputs and servers maintain proprietary models. One of their contributions is the development of a custom method for matrix multiplication based on homomorphic encryption.

Adams et al. [92] introduce the first application of MPC protocols to text classification using CNN. Their method adapts a CNN-2D from the Crypten framework into a CNN1D, using two 2D convolutional layers to emulate the behavior of a CNN1D. In contrast, this research directly implements a one-dimensional and private EL, resulting in a more efficient solution. Furthermore, while [92] focuses on word-level classifications, this research emphasizes secure character-level text classification. Finally, Adams et al. [92] do not address private embeddings, assuming that *Alice* converts her text into an embedding vector using a public BERT model before sharing it secretly. This method is not feasible when the embedding is part of *Bob*'s confidential information. This work overcomes this limitation by providing protocols and implementations for secure embeddings.

The SecureNLP model [93] presents two security protocols for LSTM and RNN in the honest but curious model. The main difference from this research is that it uses LSTM for character-level inference, while SecureNLP performs word-level inference. Additionally, SecureNLP does not support private EL.

Knott et al. [94] provides a comprehensive overview of the Crypten framework, demonstrating its application in text classification, speech recognition, and image classification. Their work performs text classification through a sentiment analysis experiment using a linear layer applied over word embeddings. Their approach differs from this research, focusing on word-level rather than character-level classification. They do not include private embeddings or provide protocols and implementations for LSTM.

3.4 IDS with Privacy-Preserving Machine Learning

Oliveira et al. [4] propose F-NIDS, which is a network IDS based on FL for IoT networks. It uses FL and DP techniques with asynchronous communication between system entities, aiming for scalability and data confidentiality. F-NIDS is adaptable to cloud and fog IoT environments, maintaining satisfactory attack detection and classification performance. The study evaluates three robustness strategies: Gaussian noise adjustment to protect against inference attacks, sample size variation to secure training data, and resilience against model inversion attacks.

Popoola et al. [95] propose Federated Deep Learning (FDL) for IDS in heterogeneous networks. The FDL model showed better classification and generalization performance than local DNN models. Furthermore, the Fed+ algorithm outperformed two state-of-the-art fusion algorithms: federated averaging (FedAvg) and coordinated averaging (CM).

Nguyen et al. [96] provides a comprehensive review of emerging applications of FL in IoT networks, covering recent advances in FL and IoT to their integration.

Zhu et al. [68] propose a scheme that combines perturbation encryption and data encryption to protect privacy, using k-Nearest Neighbors (kNN) as the detection algorithm.

Celdrán et al. [97] propose a novel host-based, federated learning-driven IDS for IoT spectrum sensors, employing ML and unsupervised deep learning, as well as system call-based fingerprinting.

Aouedi et al. [98] introduce a IDS framework focused on federated mixture models applied to IoT and Industrial IoT (IIoT). The framework initially uses primary classifiers, such as Decision Tree (DT) and Random Forest (RF), to generate metadata, and the Neural Network (NN) metaclassifier uses this metadata in the federated training step to perform the final classification on the test set. Metadata, rather than sensitive user data, is used in the training of the federated metaclassifier to enhance privacy.

Neto et al. [99] examine the correctness, validity, usefulness, and data collection pipeline of the IoT datasets for experiments.

Li et al. [100] proposes the Dynamic Weighted Aggregation Federated Learning (DAFL) IDS based on FL. Compared to conventional IDS, DAFL implements dynamic filtering and weighting strategies for local models, improving IDS with lower communication overhead.

Attota et al. [101] present the MV-FLID approach, which uses FL on multiple views of IoT network data to detect and classify attacks in a decentralized manner, maximizing the detection efficiency of different classes of attacks.

Folino et al. [102] investigate data mining algorithms applied to IDS in distributed and cloud environments, focusing on solutions based on the ensemble paradigm.

The works of [2,103–107] involve the use of FL in IDS, with [106] using a MPC approach. In contrast, [104] focuses on an IDS approach for the IoT, and [103] applies it to mobile devices.

Truex et al. [108] shows that the same global DP could be guaranteed by adding a "fraction" of DP noise locally rather than using MPC protocols or the proposed DP protocol.

3.5 Attacks and Defenses on Machine Learning

Carlini et al. [21] propose an analysis of membership inference attacks, which allow an adversary to identify whether a specific piece of data was present in the training set of a machine learning model. It argues that the evaluation of these attacks should focus on the true positive rate at low false positive rates ($\leq 0.1\%$) since commonly used average accuracy metrics do not capture the confidence of the attack in identifying training set members. To this end, the authors develop the Likelihood Ratio Attack (LiRA).

Sánchez et al. [109] discusses FL offers a solution by preserving privacy, but it is vulnerable to malicious actors. This work contributes with a new dataset suitable for FL, modeling the behavior of spectrum sensors under different SSDF attacks, and with experiments that evaluate the robustness of the federated models against three families of sensors, eight SSDF attacks, four detection scenarios (supervised and unsupervised), up to 33% of malicious participants, and four anti-adversarial aggregation functions.

Fredrikson et al. [110] introduce a new model inversion attack class that exploits prediction confidence values to extract sensitive information. The attacks have been applied to decision trees for lifestyle surveys and neural networks for facial recognition, allowing them to infer sensitive information and reconstruct facial images. In addition, researchers have investigated countermeasures such as training decision trees with privacy and rounding confidence values, showing how to prevent these attacks with minimal utility loss.

Mansouri et al. [111] examine the suitability of secure aggregation based on cryptographic schemes for FL, formally defining the problem and categorizing existing solutions. In addition to exploring the specific challenges of FL, it reviews recent secure aggregation solutions. It proposes an improved definition of secure aggregation that better fits this context while identifying future research directions.

Nguyen et al. [112] presents FLAME, a defense framework that estimates the amount of noise needed to eliminate backdoors. It uses model clustering and weight pruning to minimize injected noise. The evaluation of FLAME on image classification, word prediction, and IoT intrusion detection datasets demonstrates that it effectively removes backdoors with minimal impact on benign model performance.

Bonawitz et al. [113] present a communication-efficient and fault-robust protocol for secure aggregation of high-dimensional data. The protocol allows a server to securely compute the sum of large data vectors maintained by users on mobile devices (without learning the individual contributions). It can be used, for example, in federated learning to aggregate user-contributed deep neural network model updates. This work proves the protocol's security in honest-but-curious and active adversary scenarios. This work shows that security is maintained even if an arbitrary subset of users abandons the process at any time.

Haoyang et al. [114] propose 3DFed, an adaptive, extensible, and multi-layered framework for performing covert backdoor attacks in a black-box scenario. 3DFed has three evasion modules that camouflage the backdoor models: training with constrained loss, noise mask, and decoy model. By implanting indicators into the backdoor model, 3DFed obtains feedback from the global model and dynamically adjusts the hyperparameters of the evasion modules.

Shejwalkar et al. [115] propose to develop a systematization of poisoning attacks in FL, enumerating possible threat models, poisoning variations, and adversary capabilities. This work focuses on non-targeted poisoning attacks, which are highly relevant for FL deployments in production. This work critically analyzes these attacks in practical FL environments, carefully characterizing the threat models and realistic adversary capabilities. This work proposes data and model poisoning attacks and demonstrates, through experiments on three benchmark datasets, the (in)effectiveness of poisoning attacks in the presence of simple defenses.

3.6 Frameworks PPML

This section will list some frameworks used and comment on the need to unite them all with Privacy-Preserving Machine Learning Framework (P^2MLF).

Keller [19] proposes MP-SPDZ, which is software designed to evaluate the performance of several MPC protocols under different security models, including honest or dishonest majority and semi-malicious (passive) and malicious (active) corruption scenarios. It incorporates technologies such as secret sharing, Homomorphic Encryption (HE), and garbled circuits to support various security configurations. P^2MLF build MPC protocols using MP-SPDZ and perform inference with trained models such as CNN1D, LSTM, MLP, EL, and Logistic Regression (LR). An MPC protocol is also employed to aggregate models trained with FL. In addition, MPC is used for training LR, allowing the inclusion of Laplacian noise after training to create a differentially private LR model.

Beutel et al. [116] propose Flower is a framework that adapts existing ML algorithms and training pipelines to federated settings, allowing the evaluation convergence properties and training times. It supports extensions for mobile and wireless clients with varying compute, memory, and network resources and simulates these conditions in cloud environments for realistic testing. Designed for scalability, Flower allows connecting and training large numbers of clients simultaneously, providing a platform for FL experiments. Flower does not support MPC protocols, as its focus is to act as a framework for FL. Integrating protocols such as MPC or HE could increase privacy by allowing aggregators to aggregate local models in one global model. Alternatively, improve security and privacy by aggregating and detecting adversarial attacks on encrypted data. P^{2}MLF build FL protocols using Flower.

Google Research team proposes TensorFlow Privacy (TF-Privacy) [1], an open-source library that provides implementations of optimizers with DP support for training ML models using DP-SGD. P^{2}MLF trains EL, CNN1D, LSTM, and MLP models using DP-SGD from the TF-Privacy library, with DP optimizers provided by the TensorFlow and Keras APIs.

Ziller et al. [117] propose PySyft, which is an open-source library that improves security and Privacy-Preserving Machine Learning (PPML) by integrating with popular frameworks such as PyTorch, providing a user-friendly interface for researchers to implement FL, MPC and DP methods. Designed for accessibility and extensibility, PySyft allows researchers to explore privacy-preserving techniques easily.

Like PySyft the P^{2}MLF framework supports MPC, FL, and DP. However, the difference is that PySyft does not contain native MPC protocols for EL, LSTM, and CNN1D built on MP-SPDZ [19] to the P^{2}MLF. P^{2}MLF uses MP-SPDZ because it contains many protocols MPC for performing experiments.

[1]`https://github.com/tensorflow/privacy`

Chapter 4

P^{2}MLF

This chapter presents Privacy-Preserving Machine Learning Framework (P^{2}MLF) [1] proposal to ensure data privacy in Machine Learning (ML).

The framework focuses on mechanisms for performing inference and training with data privacy guarantees.

This work refers to techniques that use training data from multiple parties as collaborative training, as these parties work together to develop a ML model using data from all participants. Therefore, methods using Secure Multi-Party Computation (MPC) and Federated Learning (FL) will be classified as collaborative training.

P^{2}MLF is applied to the problem Domain Generation Algorithms (DGA) (see Chapter 6) using all framework phases: preprocessing, training, model, and inference. In terms of training, only the non-collaborative training method uses Differential Privacy (DP) guarantees through the use of Differentially-Private Stochastic Gradient Descent (DP-SGD).

For naming purposes, this work refers to P^{2}MLF training as secure training and P^{2}MLF inference as secure inference.

The solution incorporates collaborative training to address the Collaborative Intrusion Detection Systems (CIDS) problem (see Chapter 5).

4.1 Introduction

Figure 4.1 outlines an application life cycle's different phases incorporating ML, including preprocessing, training, modeling, and inference. In addition, it demonstrates how to map these phases to solutions that ensure privacy.

Although designed for DGA and CIDS problems, P^{2}MLF allows adaptation to other issues by reusing parts of its structure. For example, it is possible to perform training without privacy but use the trained model in the clear in a method with secure inference.

[1]`https://github.com/ricardojmmaia/p2mlf`

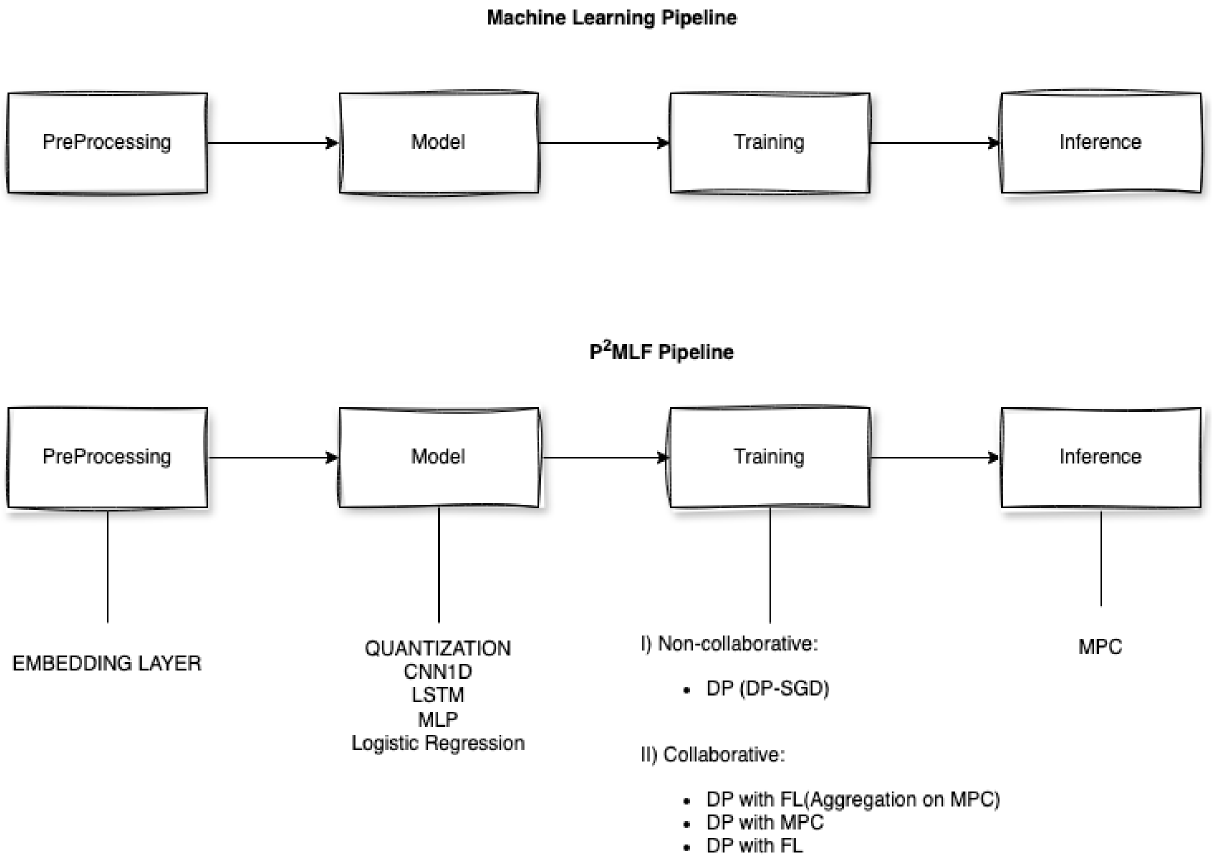

Figure 4.1: P^{2}MLF pipeline

The proposed block-based overview lets the engineer understand which privacy requirements must follow each phase of a ML pipeline. In this case, an engineer or designer looking to use P^{2}MLF for other problems must consider the trade-off between the privacy level and runtime in the case of collaborative training, as there is the possibility of using one of the following approaches: DP with FL (aggregator using MPC), DP with MPC, and DP with FL.

For a simple explanation of the phases of P^{2}MLF, this work will assume that *Alice* is the data owner and *Bob* is the model owner. In the case of collaborative training, where the goal is to perform model training using data from different parties, this thesis will introduce *Clara* and *Dan* while also considering that *Bob* also intends to use his data for training.

Figure 4.2 shows two main aspects of the P^{2}MLF that provide security and inference, keeping the privacy data; there is a proposal to preprocessing in the general training and inference are two main aspects in ML.

Consider the inference represented in the diagram 4.2, where *Alice* has input data and the desire to get inference without providing her clear data. Consider clear data as original information without protection from cryptographic protocols.

The proposal is to provide secure inference to *Alice* and keep the privacy of Bob's model parameters. The P^{2}MLF uses MPC protocols to create secure inference, where the MPC protocols will detailed in the section 4.5, and 4.6.

P^{2}MLF focused only on the MPC protocols necessary to the applications DGA(see Chapter 6) and CIDS (see Chapter 5). Creating MPC protocols to represent the respective ML model will be necessary if an engineer intends to use secure inference.

Regarding the secure training on P^{2}MLF, four proposals are divided into collaborative and non-collaborative trains. The non-collaborative training is a situation where only *Bob* has the training data. Collaborative training is a situation where only *Bob*, *Clara*, and *Dan* intend to create a ML model. The privacy collaborative training is beneficial in situations where *Bob* needs another training data (*Clara* and *Dan* in this scenario) to detect different cases and improve the accuracy of the *Bob*'s model.

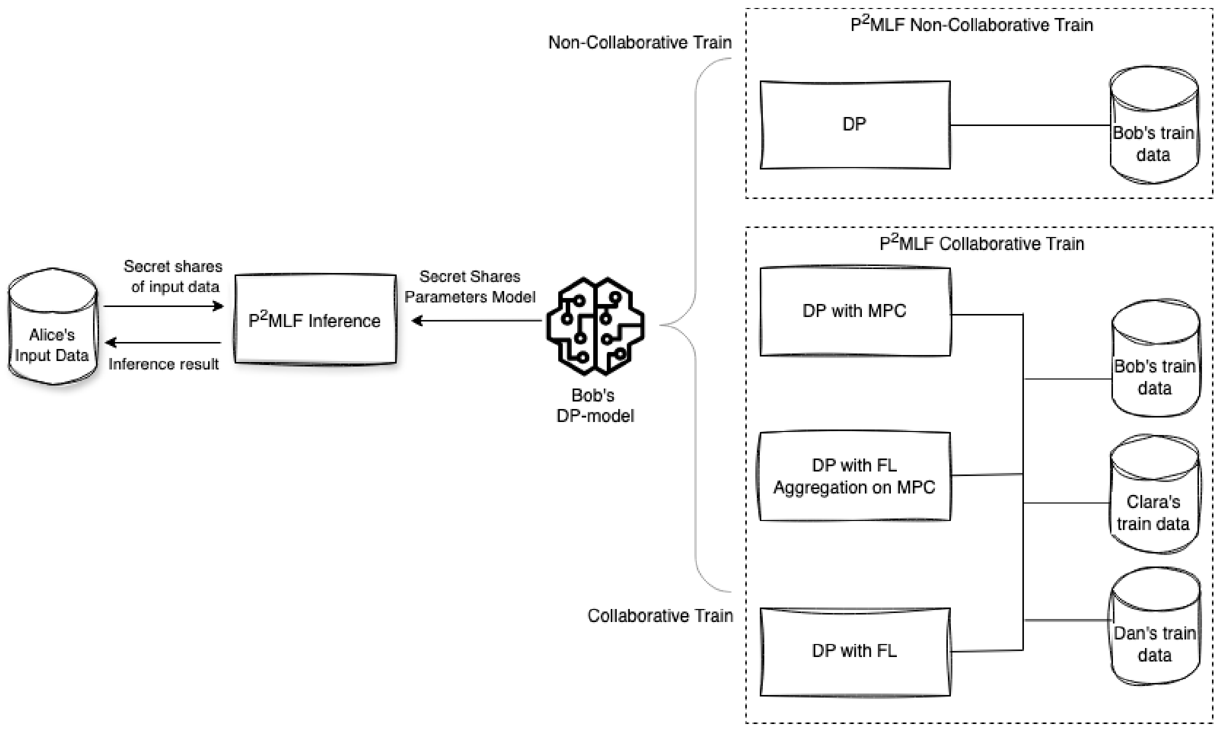

Figure 4.2: P^{2}MLF Framework Overview

4.2 Threat Model

P^{2}MLF builds the MPC protocols on existing MPC primitives [19]. It is possible to adapt the proposed protocols to any threat model by replacing the underlying primitives with those available in the literature for the specified threat model. This work demonstrates how the protocols work for the case of "honest-but-curious" servers, which are participants that follow the protocol instructions but try to obtain as much information as possible about the secret data. This work considers threat models:

- With two and three computing servers.
- That withstands attacks by any adversary that can corrupt one server that he chooses, meaning the system preserves privacy guarantees even if one of the MPC servers becomes corrupted.

4.3 Privacy Requirements

Bob cannot learn anything about *Alice's* input. *Alice* should only learn the result of the classification protected by (ϵ, δ)-DP and cannot learn anything else about *Bob's* model and training data. The MPC servers cannot learn anything about *Alice's* and *Bob's* private inputs.

P^{2}MLF supports the input and output privacy requirements. Regarding output privacy, the goal is to protect the data used to perform model training because, only with the input privacy requirement, it is possible for *Alice* (considering P^{2}MLF Inference) to get information of the *Bob*'s model during secure inference. To ensure privacy in the output, P^{2}MLF uses the guarantees of DP during the training of the models.

In the training with privacy requirements, the P^{2}MLF guarantees input privacy, where only the owner data knows their data, on MPC protocols. In the case of collaborative training, *Bob*, *Clara*, and *Dan* do not need to share their data with a trusted third party. In addition, no other party will know the data of another party. Furthermore, P^{2}MLF ensures that during collaborative training, there is no leakage of data from *Bob*, *Clara*, and *Dan*.

4.4 Basic Building Blocks

This work builds on the proposed protocols on a few existing building blocks in the MP-SPDZ framework [19]. This work uses MPC protocols π_{SIGMOID} for the sigmoid function, π_{SOFTMAX} for the softmax function, π_{MUL} for the secure dot product, π_{TANH} for the hyperbolic tangent, π_{RELU} for the relu function, π_{DENSE} for the dense layer, and π_{LR} for Logistic Regression (LR) model. See [19] for a detailed description of these primitives.

4.5 P^{2}MLF Preprocessing

This section presents a proposal to ensure privacy in preprocessing.

Considering the requirement that only the owner knows their data, *Bob* does not want to pass their model, which is their data, to *Alice*.

Therefore, in the DGA problem, *Alice* needs a MPC protocol to represent Embedding Layer (EL) to process their text input Domain Name System (DNS) into numbers.

P^{2}MLF privately obtains the embeddings, which are numerical values, and then passes them to the subsequent layers (One-Dimensional Convolutional Neural Network (CNN1D), Long Short-Term Memory (LSTM), or MultiLayer Perceptron (MLP)) for the DGA application.

The question is whether there is any preprocessing of *Alice*'s data that depends on any information from *Bob*. An MPC protocol should represent the desired preprocessing algorithm. This work requires an EL for the reasons outlined in this section's beginning.

It is possible to structure the pipeline in blocks and add new MPC protocols depending on the problem; it is essential to consider the case where there is processing with Alice's data that depends on some data from Bob.

4.5.1 Embeddings

The EL transforms the input matrix $\mathbf{X}$ from *Alice* to provide a vector representation of the characters in the input text using *Bob's* learned embedding weights matrix $\mathbf{W}$.

Many previous works for MPC-based privacy-preserving text classification, including DGA detection, require *Alice* to embed the input text, which in turn requires the trained embeddings to be made public and may leak information regarding training data unless trained with DP guarantees [18, 92]. Moreover, these trained embeddings may be proprietary.

In order to mitigate the problems related to the above scenarios, this work proposes a novel MPC protocol for the embedding layer to compute the embeddings of private input text in an oblivious manner. The vectors resulting from the embedding layer of the classification models represent the lexical information of the characters in the given DGA domain (URL) in the ASCII character set. The idea behind $\pi_{\mathsf{EMBEDDING}}$ is simple: this work extracts the vector representation of each character in the input text (in this case, it is a domain or URL) from the trained embeddings with guarantees of DP.

One of the simplest ways to extract such embeddings (Protocol 1) is to represent the input text as one hot encoded matrix $[\![\mathbf{X}]\!]$ of dimension $l \times 128$ and multiply it with the weights of trained embeddings $[\![\mathbf{W}]\!]$ of dimension 128×128. The product is a matrix $[\![\mathbf{Y}]\!]$ of dimension $l \times 128$ that represents a set of vector representations of each character in the input text. This work notes that feature extraction done this way requires only multiplication operations for which state-of-the-art MPC primitives are available, resulting in optimized performance of the MPC protocol to extract embeddings of the input. Moreover, this protocol is general enough to work with character sets of arbitrary cardinality c.

Protocol 1: $\pi_{\mathsf{EMBEDDING}}$ for secure inference of embedding layer

```
Input  : Shared secret matrices $[\![\mathbf{X}]\!]$ of dimension $l \times c$ representing one-hot encoded
         inputs, and matrix $[\![\mathbf{W}]\!]$ of dimension $c \times d$ representing embedding weights
         are used, considering $l$ the length of the input text in characters, $c$ as the
         cardinality of the character set and $d$ is the dimensionality of the embedding
         space.
Output : A shared secret embedding matrix $[\![\mathbf{Y}]\!]$ of dimensions $l \times d$ is generated from
         the input domain to be classified.
1 $[\![\mathbf{Y}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\mathbf{X}]\!], [\![\mathbf{W}]\!])$
2 return $[\![\mathbf{Y}]\!]$
```

This work provides the first protocol and implementation for an embedding layer in MPC and implements it in the MP-SPDZ framework. The main idea behind this solution is to represent *Alice*'s input to the protocol as a matrix $[\![\mathbf{X}]\!]$ where each row of $[\![\mathbf{X}]\!]$ is one hot encoding of a character of the domain name to be classified. So, each row of $[\![\mathbf{X}]\!]$ consists of a binary vector where precisely one bit sets to 1. An inner product is made between $[\![\mathbf{X}]\!]$ and the private embedding matrix $[\![\mathbf{W}]\!]$, resulting in $[\![\mathbf{Y}]\!]$, and the result of the embedding layer is the input data for the CNN1D, LSTM, and MLP models [3].

4.6 P^{2}MLF Models

This section shows MPC protocols built to represent a ML model needed for P^{2}MLF inference.

The challenge inherent to each of the models created in this section concerns the restrictions imposed by MPC that restrict the use of addition and multiplication operations. An engineer who wishes to use P^{2}MLF for inference must create an MPC protocol, if one does not exist, to represent the model or layers of an ML model.

4.6.1 MLP

This work leverages an existing implementation of an MPC protocol for secure inference with a MLP, available within the MP-SPDZ framework [19]. MLPs represented by π_{DENSE} will be used as a baseline method in this framework.

The input is secret shared by *Alice*, while *Bob* secret shares the model weights. *Bob*'s model in the architecture with MLP composes the weights of the embedding and dense layers.

4.6.2 CNN1D

This solution is now presented based on CNN1D. This architecture has an input layer with protocol $\pi_{\mathsf{EMBEDDING}}$, a layer with protocol π_{CNN1D}, and a layer with protocol π_{DENSE} representing the dense layer in MPC. The input will be secret shared by *Alice*, while *Bob* secret-shares the model's weights.

This work leverages a proposal for a CNN1D [92]. The protocol π_{CNN1D} for secure inference with CNN1D uses (a) the existing MPC protocols available in the literature for π_{RELU}, and π_{MUL} (b) the MPC protocols for embedding is $\pi_{\mathsf{EMBEDDING}}$ and CNN1D is π_{CNN1D}.

In Protocol 2 for secure inference with a CNN1D layer, it is used as input of the secret shared embeddings obtained as output from the $\pi_{\mathsf{EMBEDDING}}$ protocol and the parameters of the secret shared model, i.e. the weights of the kernel (k in total), each of size k for the CNN1D layer of *Bob*.

Protocol 2: π_{CNN1D} for secure inference with CNN1D Convolution

```
Input  : The shared secret matrices $[\![\mathbf{X}]\!]$, obtained as the output of the private
         embedding computed by the $\pi_{\mathsf{EMBEDDING}}$ Protocol, $[\![b]\!]$, and $[\![\mathbf{w}]\!]$ representing
         input, bias, and weight as secret shared. The constants $k$ and $l$ represent the
         kernels and rows of $[\![x]\!]$.
Output : A secret shared $[\![\mathbf{y}]\!]$ of dimension $l - k + 1$.
1 for $i \leftarrow 0$ to $l - k + 1$ do
2     $[\![\mathbf{y}[i]]\!] \leftarrow [\![b]\!]$
3     for $j \leftarrow 0$ to $k - 1$ do
4         $[\![\mathbf{y}[i]]\!] \leftarrow [\![\mathbf{y}[i]]\!] + \pi_{\mathsf{MUL}}([\![\mathbf{X}[i+j]]\!], [\![\mathbf{w}[j]]\!])$
5     end
6 end
7 return $\pi_{\mathsf{RELU}}([\![\mathbf{y}]\!])$
```

4.6.3 LSTM

Now, the solution based on LSTM includes an input layer with the $\pi_{\mathsf{EMBEDDING}}$ protocol, a layer with the π_{LSTM} protocol, and a layer with the π_{DENSE} protocol representing the dense layer in MPC. *Alice* securely shares her input, while *Bob* securely shares the model weights.

Protocol 3 describes the layer LSTM. The input for the LSTM layer is the secret shared output of the EL, while the *Bob* secret shares the kernel weights ($[\![w_f]\!], [\![w_i]\!], [\![w_o]\!], [\![w_c]\!]$) and the biases ($[\![b_f]\!], [\![b_i]\!], [\![b_o]\!], [\![b_c]\!]$). This work refers the reader to Section 2.2.3 for an explanation of these terms. Operations involve MPC protocols π_{SIGMOID} for the sigmoid, π_{MUL} for secure multiplications, and π_{TANH} for the hyperbolic tangent.

Protocol 3: π_{LSTM} for secure LSTM

Input : Secret shared vector x (obtained as output of the private embedding computed by Protocol $\pi_{\mathsf{EMBEDDING}}$), and secret shared values of the weights ($[\![w_f]\!]$, $[\![w_i]\!]$, $[\![w_o]\!]$, $[\![w_c]\!]$), and biases ($[\![b_f]\!]$, $[\![b_i]\!]$, $[\![b_o]\!]$, $[\![b_c]\!]$). Let $[a, b]$ be the concatenation of a and b, with the input length publicly known.

Output : In the inference process the result after the LSTM layer is a secret shared vector $[\![y]\!]$.

```
1  $[\![h_0]\!] \leftarrow 0$
2  $[\![c_0]\!] \leftarrow 0$
3  for $t \leftarrow 1$ to $n$ do
4      $[\![f_t]\!] \leftarrow \pi_{\mathsf{SIGMOID}}(\pi_{\mathsf{MUL}}([\![w_f]\!], [[\![h_{t-1}]\!], [\![x[t]]\!]]) + [\![b_f]\!])$
5      $[\![i_t]\!] \leftarrow \pi_{\mathsf{SIGMOID}}(\pi_{\mathsf{MUL}}([\![w_i]\!], [[\![h_{t-1}]\!], [\![x[t]]\!]]) + [\![b_i]\!])$
6      $[\![c_t']\!] \leftarrow \pi_{\mathsf{TANH}}(\pi_{\mathsf{MUL}}([\![w_c]\!], [[\![h_{t-1}]\!], [\![x[t]]\!]]) + [\![b_c]\!]))$
7      $[\![c_t]\!] \leftarrow \pi_{\mathsf{MUL}}([\![f_t]\!], \pi_{\mathsf{MUL}}([\![c_{t-1}]\!], \pi_{\mathsf{MUL}}([\![i_t]\!], [\![c_t']\!])))$
8      $[\![o_t]\!] \leftarrow \pi_{\mathsf{SIGMOID}}(\pi_{\mathsf{MUL}}([\![w_o]\!], [[\![h_{t-1}]\!], [\![x[t]]\!]]) + [\![b_o]\!])$
9      $[\![h_t]\!] \leftarrow \pi_{\mathsf{MUL}}([\![o_t]\!], \pi_{\mathsf{TANH}}([\![c_t]\!]))$
10     $[\![y[t]]\!] \leftarrow [\![h_t]\!]$
11 end
12 return $[\![y]\!]$
```

4.6.4 LR

Protocol 4 describes the protocol MPC for secure inference π_{LR} is known by *Alice* and *Bob*. *Alice* provides the input vector $\mathbf{x}$, and *Bob* provides the LR model of coefficients vector $\mathbf{w}$ and bias b. The $\pi_{\mathsf{LRINFERENCE}}$ protocol uses a MPC π_{SIGMOID} protocol after the secure sum π_{SUM} of the bias $[\![b]\!]$ and the result of secure dot product *Alice*'s input $[\![\mathbf{x}]\!]$ with *Bob*'s coefficients $[\![\mathbf{w}]\!]$. At the end step of the π_{LR} protocol, *Alice* and *Bob* receive the inference values $[\![y]\!]$. In the experiments conducted in this study, this work used a previously trained

LR model with DP. However, the $\pi_{\text{LRINFERENCE}}$ protocol allows loading LR models trained earlier with or without DP.

Protocol 4: $\pi_{\text{LRINFERENCE}}$ is the MPC protocol to secure inference of LR model

```
    Input  : secret-shared input vector $[\![\mathbf{x}]\!]$ of length $n$ represents input data, vector $[\![\mathbf{w}]\!]$ of
             lenght $n$ represents coefficients of the LR model, and $[\![b]\!]$ value represents the
             model's bias.
    Output : A secret-shared value $[\![y]\!]$ represents output of the model
1   $[\![y]\!] \leftarrow \pi_{\text{SIGMOID}}(\pi_{\text{SUM}}(\pi_{\text{DOTPRODUCT}}([\![\mathbf{x}]\!], [\![\mathbf{w}]\!]), [\![b]\!]))$
2   return $[\![y]\!]$
```

4.7 P^{2}MLF Inference

P^{2}MLF inference is represented by the Figure 4.3. To ensure the inference privacy in the P^{2}MLF uses MPC protocols.

In the secure inference, *Alice* and *Bob* send servers the secret shares of their data. In this case, *Alice* sends the secret shares of the input data, and *Bob* sends the representation of the secret shares of the parameters of the model ML. Then, the servers compute only the secret shares of the *Alice*'s input data and *Bob*'s model parameter instead of clear data. The servers compute secret shares in the final phase step and then return the inference result.

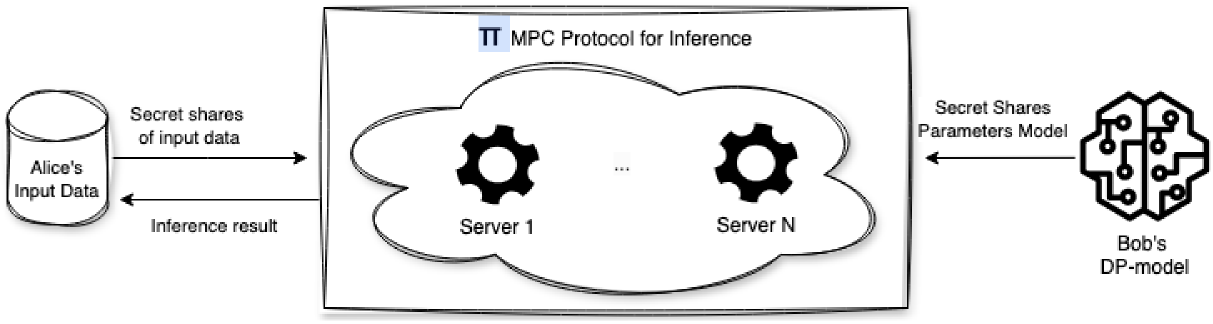
π MPC Protocol for Inference
Secret shares of input data
Alice's Input Data
Inference result
Server 1
...
Server N
Secret Shares Parameters Model
Bob's DP-model

Figure 4.3: P^{2}MLF Inference

4.8 P^{2}MLF Training

This section discusses four methods to secure training, divided into non-collaborative and collaborative training. Regarding collaborative training, there are methods proposals summarized in figure 4.4: DP with FL; DP with MPC; DP with FL using aggregations of model parameters with MPC;

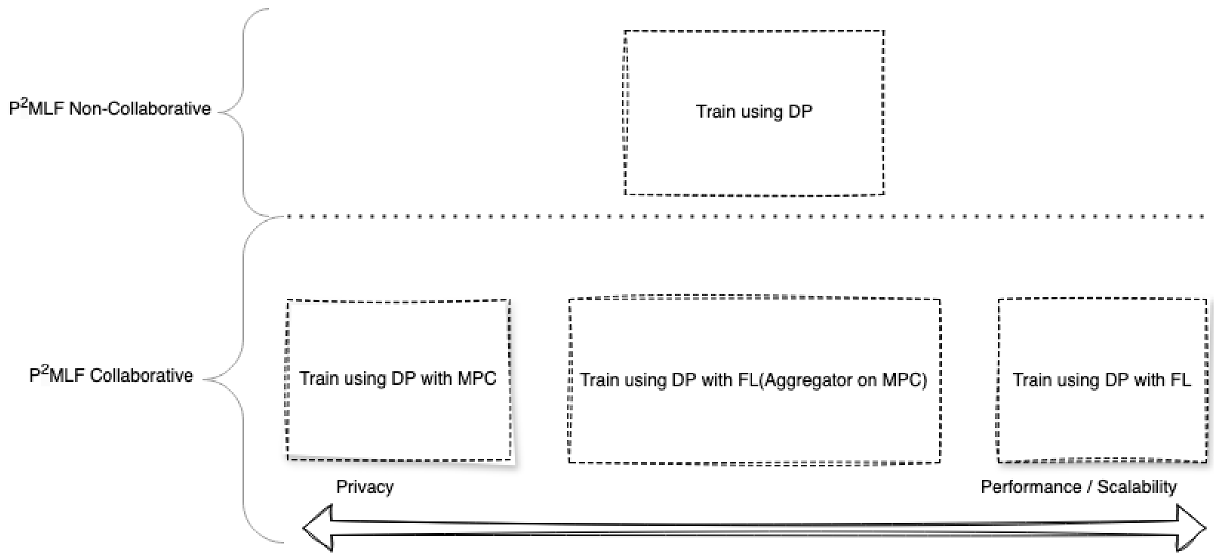

Figure 4.4: P^{2}MLF Training

4.8.1 Basic Building Blocks

P^{2}MLF training establishes components within the MPC framework, specifically using MP-SPDZ, as referenced in [19] and FL Framework called Flower, as detailed in [116]. Additionally, P^{2}MLF training integrates the LR with DP on MPC protocol from the works cited in [5].

Furthermore, this section uses LR with DP as created in the work cited in [118].

P^{2}MLF inference uses MPC protocols π_{SIGMOID} for the sigmoid function, $\pi_{\mathsf{DOTPRODUCT}}$ for the secure dot product, π_{SUM} for the sum, $\pi_{\mathsf{LRINFERENCE}}$ for inference LR, π_{LR} for the training LR. See [19] for a detailed description of these MPC primitives. P^{2}MLF training uses the π_{DP} protocol to generate DP noise on MPC, described in [5].

P^{2}MLF training uses FL primitives π_{FLSERVER} for FL Server protocol, π_{FLCLIENT} for the FL Client protocol, and $\pi_{\mathsf{FLAGGREGATOR}}$ that combines local LR on each client with a server that performs model averaging LR. For a detailed description of these primitives FL, refer to [116]. In the FL protocols, this work implements the π_{LRDP} protocol to perform LR training with DP, as implemented in the work cited in [118].

4.8.2 Collaborative Training

The collaborative P^{2}MLF training is helpful in situations where *Bob* wants to improve his model using other training data and thus detect patterns belonging to the training data of *Clara* and *Dan*. However, in collaborative training, the P^{2}MLF training aims to guarantee privacy in both input and output. The framework P^{2}MLF can be used in collaborative training with n clients.

4.8.2.1 DP with FL

Figure 4.5 represents this collaborative training method that uses protocols FL with DP.

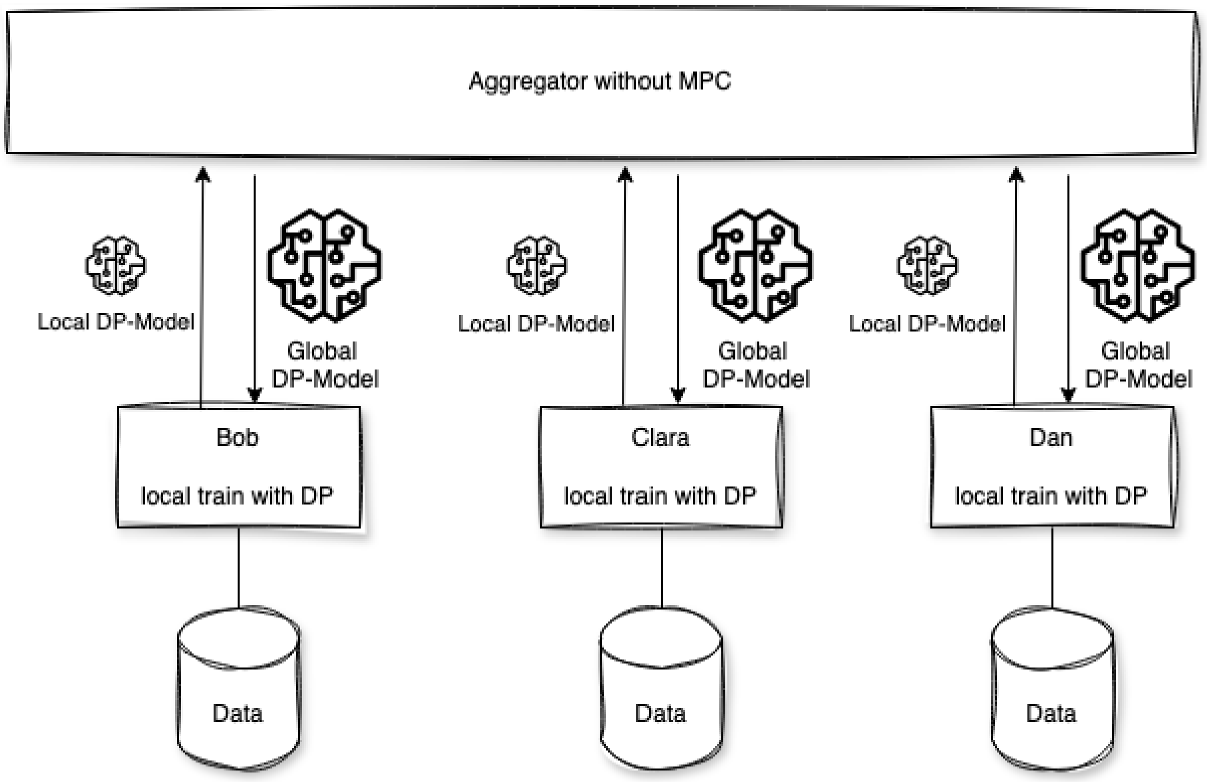

Figure 4.5: P^{2}MLF Collaborative Training using DP with FL

This method describes the FL training protocol using two protocols, π_{FLSERVER} represented by protocol 6 and π_{FLCLIENT} represented by protocol 5. Each client possesses their data and performs local training with it. Each party's data, split horizontally, consists of a matrix with input data $\mathbf{X}$. The function LR_train_{dp}, which embodies training with LR and DP [118].

The expression $\mathbf{w}_i{}^e \leftarrow \mathbf{w}^e$ means that each model of each client i at epoch e receives the latest global model.

$\mathbf{w}_i{}^{e+1} \leftarrow LR\_train_{dp}(\mathbf{w}_i{}^e, \mathbf{X})$ means that the next client local model will be trained in 1 epoch using the local data of client i.

Finally, the local model $\mathbf{w}_i{}^{e+1}$ will be sent to the FL server to be aggregated with other local models from other clients.

After clients send their models to the server, the $\pi_{\mathsf{FLAGGREGATOR}}$ protocol aggregates them into a single global model using an arithmetic mean, as detailed in [119].

In this method, the aggregator receives the model's parameter in the clear, and this could be a security failure point because it aggregates models from different parties without privacy guarantees.

Protocol 5: π_{FLCLIENT} represents the FL client in the P^{2}MLF training using LR model

```
Input  : Each client $i$ provides the matrix input data $\mathbf{X}$. The function $LR\_dp$ is the
         LR algorithm to perform the training with DP locally in each client. $\mathbf{w}_i{}^e$
         represents the model of the client $i$ trained with LR and DP in epoch $e$. $\mathbf{w}^e$ is
         the global model in the epoch $e$
Output : Each client $i$ sends to server the model $\mathbf{w}_i{}^e$
1 $\mathbf{w}_i{}^e \leftarrow \mathbf{w}^e$
2 $\mathbf{w}_i{}^{e+1} \leftarrow LR\_train_{dp}(\mathbf{w}_i{}^e, \mathbf{X})$
3 return $\pi_{\mathsf{FLCLIENT}}$ sends $\boldsymbol{w}_i{}^{e+1}$ to the FL Server
```

Protocol 6: π_{FLSERVER} is the FL Server protocol to P^{2}MLF training of LR model

```
Input  : $n$ is the total number of clients.
Output : The aggregation of the LR model's parameter of all $n$ clients. $\mathbf{w}^{e+1}$ is the next
         global model
1 for each client $i$ where $1 \leq i \leq n$ do
2     $\mathbf{w}^e{}_i \leftarrow \pi_{\mathsf{FLCLIENT}}(i)$
3 end
4 $\mathbf{w}^{e+1} \leftarrow \frac{\sum_{i \leftarrow 0}^{n} \mathbf{w}_i{}^e}{n}$
5 return $\boldsymbol{w}^{e+1}$
```

4.8.2.2 DP with FL(Aggregation on MPC)

Figure 4.6 represents this collaborative training method that uses LR with DP as created in the work cited in [118].

In this case, the client does not send the parameters in the clear to an aggregator server, but rather the secret shares of the model parameters. The aggregator server uses a MPC protocol to obtain the sum of each of the parameters of the client models. For performance reasons of the MPC protocol, the division occurs on the client side in the clear; after all, a division is computationally expensive in MPC.

For a detailed description of these primitives FL, refer to [116]. In the FL protocols, this work implements the π_{LRDP} protocol to perform LR training with DP, as implemented in the work cited in [118].

Protocol 7: $\pi_{\mathsf{SECUREFLCLIENT}}$ represents the Secure FL Client in the P^{2}MLF training using LR model

```
Input  : Each client $i$ provides the matrix input data $\mathbf{X}$. The function $LR\_dp$ is the
         LR algorithm to perform the training with DP locally in each client. $\mathbf{w}_i{}^e$
         represents the model of the client $i$ trained with LR and DP in epoch $e$. $\mathbf{w}^e$ is
         the global model in the epoch $e$. Consider that $[\![\mathbf{w}_i{}^e]\!]$ is the secret shares of $\mathbf{w}_i{}^e$
Output : Each client $i$ sends to the Secure FL Server the secret shared $[\![\mathbf{w}_i{}^e]\!]$ model
1 $\mathbf{w}_i{}^e \leftarrow \mathbf{w}^e$
2 $\mathbf{w}_i{}^{e+1} \leftarrow LR\_train_{dp}(\mathbf{w}_i{}^e, \mathbf{X})$
3 return $\pi_{\mathsf{SECUREFLCLIENT}}$ sends $[\![\boldsymbol{w}_i{}^{e+1}]\!]$ to the Secure FL Server
```

Protocol 8: $\pi_{\mathsf{SECUREFLSERVER}}$ is the Secure FL Server protocol to P^{2}MLF training of LR's model using MPC protocol to aggregate the clients's models

```
Input  : $n$ is the total number of clients.
Output : The aggregation of the LR model's parameter of all $n$ clients. $[\![\mathbf{w}^{e+1}]\!]$ is the
         next global model
1 for each client $i$ where $1 \leq i \leq n$ do
2     $[\![\mathbf{w}_i{}^e]\!] \leftarrow \pi_{\mathsf{SECUREFLCLIENT}}(i)$
3 end
4 $[\![\mathbf{w}^{e+1}]\!] \leftarrow \frac{\pi_{\mathsf{SUM}}([[\![\mathbf{w}_0]\!], \cdots [\![\mathbf{w}_n]\!]])}{n}$
5 return $[\![\boldsymbol{w}^{e+1}]\!]$
```

This training uses FL primitives $\pi_{\mathsf{SECUREFLSERVER}}$ for FL Server protocol, $\pi_{\mathsf{SECUREFLCLIENT}}$ for FL Client protocol, and π_{SUM} that is a MPC protocol that combines the local LR model on each client with a server that performs the sum of the model LR. Each client sends secret shares representing the model parameters to a set of servers that will process these secret shares.

In this case, the aggregator will aggregate the models from the different parties, ensuring the privacy of the model parameters.

To increase security in this specific method or opportunities to research:

- The aggregator should avoid adversarial attacks, in which one of the parties could poison the models by trying to change the parameters of their model to influence the global model. One way to avoid adversarial attacks would be a MPC protocol that checks whether the parameters of a ML model are an outlier, thus removing the poisoned model.

- A possible method to identify outliers in poisoned models is using the L2 Norm, the Euclidean Norm [120]. However, it is necessary to investigate the impact of using two models trained with differential privacy in this scenario. In this case, research about other distance calculation methods must focus on methods that perform well when mapped to MPC.

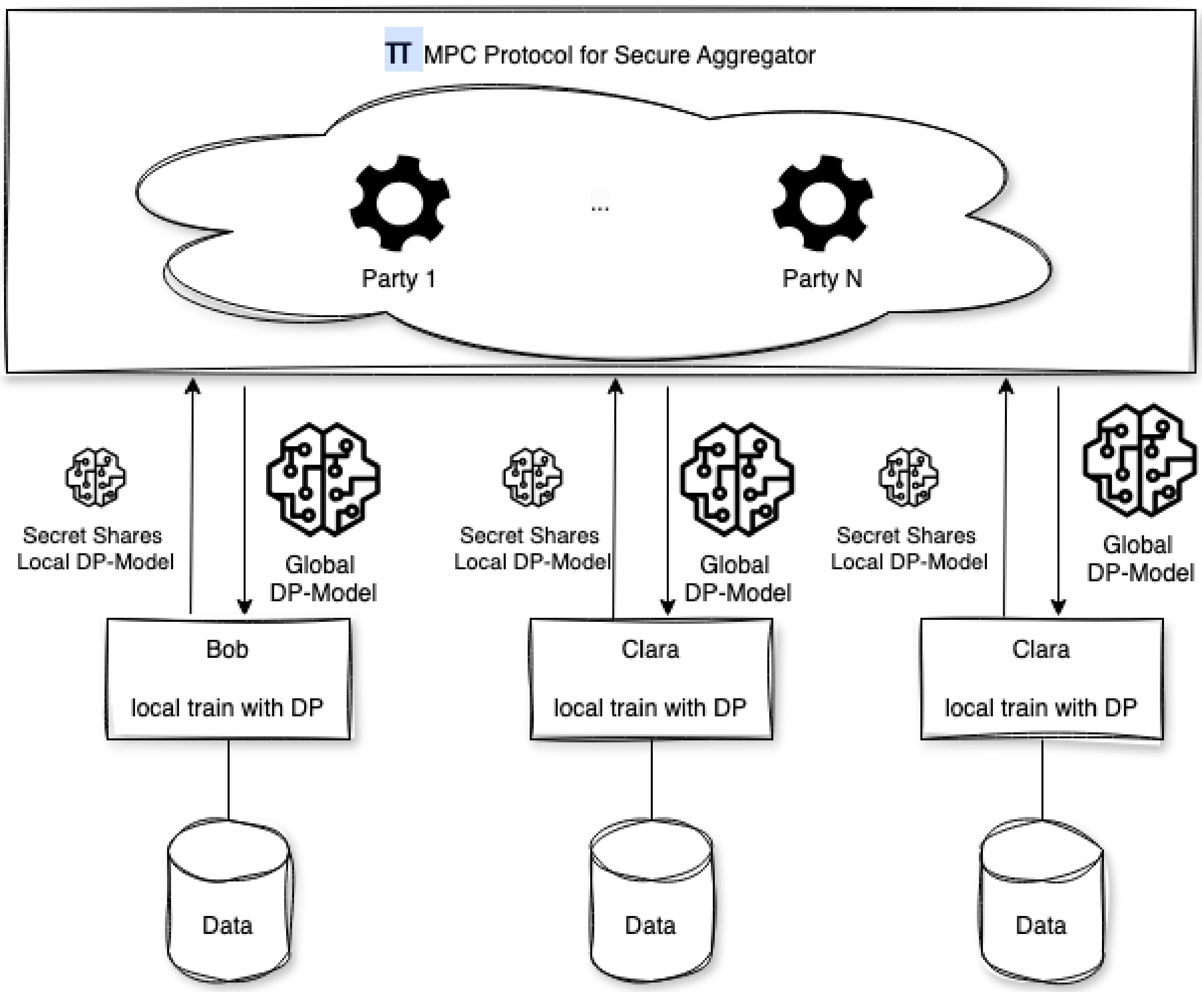

Figure 4.6: P^{2}MLF Collaborative Training using DP with FL (Aggregation with MPC)

- One point worth researching is a MPC protocol for aggregation to check if some clients use DP while training their local models.
- Another possibility is to research the impacts of a MPC aggregator protocol that adds DP generating the global model in FL scenarios.

4.8.2.3 DP with MPC

Figure 4.7 represents this collaborative training method that uses protocols MPC with DP.

For this P^{2}MLF training, this method bases the MPC protocol 9 on the work [5], which uses MPC and DP to perform the LR training. The party has horizontally partitioned the data into a matrix, including the input data x and *labels*. The first step of the protocol involves the existing MPC protocol π_{LR} to perform LR training using Stochastic Gradient

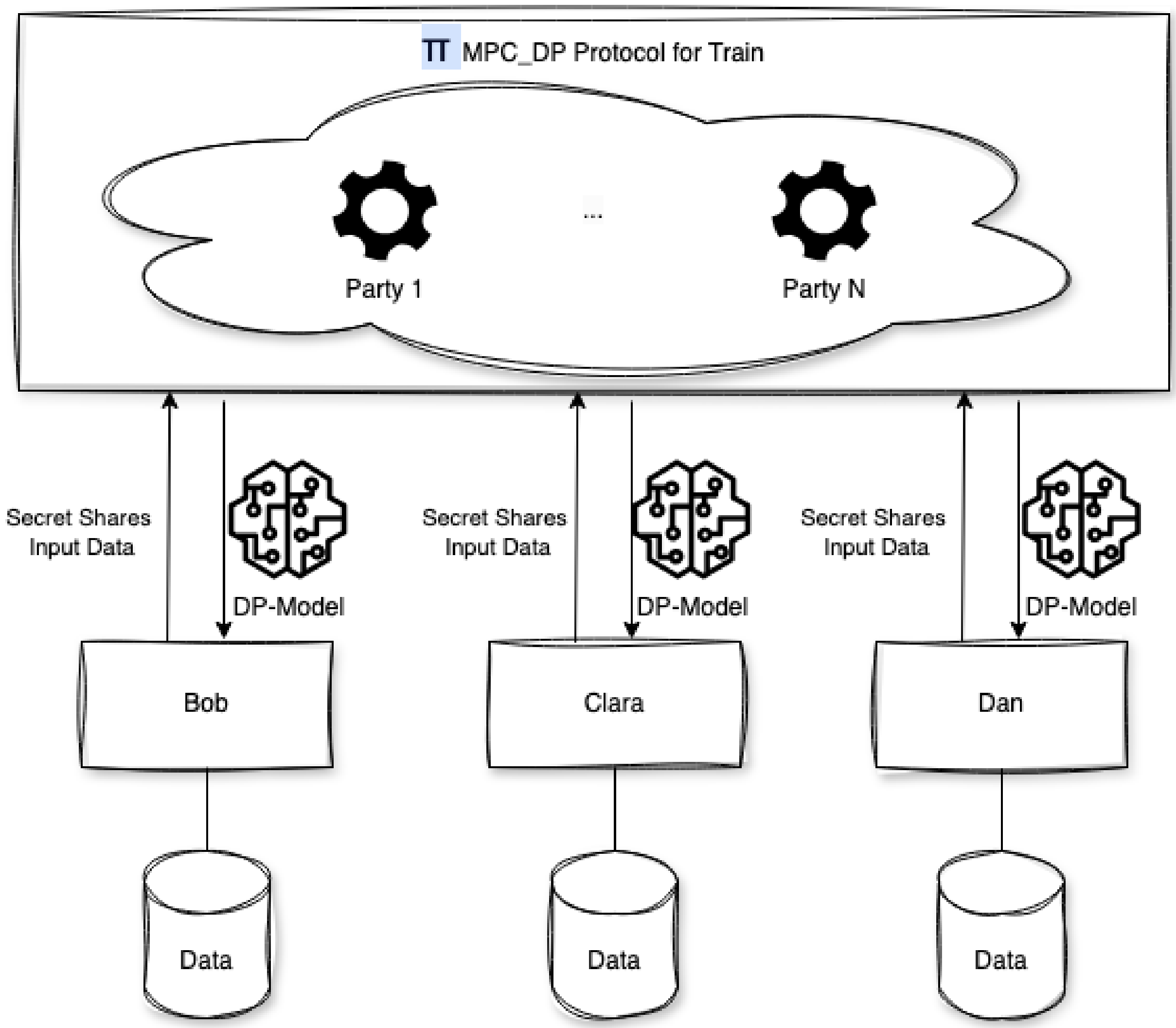

Figure 4.7: P^2MLF Collaborative Training using DP with MPC

Descent (SGD) [19]. The second step involves using the π_{DP} protocol to add Laplacian noise on the coefficients of LR model, as described in work [5].

The protocol π_{DP}, as described in [5], requires the parties to jointly generate noise appropriate for unseen model coefficients, ensuring that no entity discerns its true value. This approach safeguards the integrity of DP, allowing for the secure disclosure of noisy coefficients without violating DP's confidentiality standards [5].

This protocol $\pi_{\mathsf{LRTRAINING}}$ uses an MPC protocol to achieve ϵ by perturbing the coefficients of a trained LR model with a noise vector η. The mechanism samples noise from a probability density function defined as $h(\eta) \propto e^{-\frac{n\epsilon\Lambda}{2}|\eta|}$ [5], where n is the number of training instances, ϵ is the privacy parameter, Λ is the regularization strength, and $|\eta|$ is the L2 norm of the noise vector [5].

Two conditions must be satisfied to ensure ϵ-DP: All input feature vectors must have an

L2 norm of 1, and the LR model must use L2 regularization. When these conditions hold, the sensitivity of the coefficients is bounded by $\frac{2}{n\Lambda}$, enabling the mechanism to preserve differential privacy [5].

In summary, π_{DP} protocol defines sensitivity with the L2 norm and generates a noise vector from a multidimensional power exponential distribution. π_{DP} protocol follows three steps: First, they generate a d-dimensional Gaussian vector by sampling each coordinate from a Gaussian distribution with mean zero and variance one. Next, they normalize the vector by dividing each coordinate by its L2 norm, ensuring it is on the unit sphere. Finally, π_{DP} adjusts the vector's magnitude by multiplying each coordinate by a γ value sampled from an appropriate gamma distribution. The parties then add the resulting noise vector to the model's coefficients trained in the π_{LR} protocol, preserving privacy and ensuring compliance with DP guarantees to the LR model [5].

Protocol 9: $\pi_{\mathsf{LRTRAINING}}$ is the MPC protocol to secure training of LR model

```
Input  : Alice provides the data $[\![\mathbf{X}]\!]$ of length $n$, where the $[\![\mathbf{X}]\!]$ matrix contains input
         data. Bob provides his LR model's $[\![\mathbf{w}]\!]$ vector of length $n$ and its bias value
         $[\![b]\!]$. The LR model with DP features the coefficient vector $[\![\mathbf{w}_{dp}]\!]$ and the bias
         value $[\![b_{dp}]\!]$
1 . Output : A secret-shared value $[\![\mathbf{w}_{dp}]\!]$, and $[\![b_{dp}]\!]$ represents output of the model with
         DP
2 $[\![\mathbf{w}]\!], [\![b]\!] \leftarrow \pi_{\mathsf{LR}}([\![\mathbf{X}]\!])$
3 $[\![\mathbf{w}_{dp}]\!], [\![b_{dp}]\!] \leftarrow \pi_{\mathsf{DP}}([\![\mathbf{w}]\!], [\![b]\!])$ return $[\![\mathbf{w}_{dp}]\!]$, $[\![b_{dp}]\!]$
```

As an improvement, this protocol could incorporate the detection of adversarial attacks, considering the poisoning of the parties' data.

4.8.3 Non-Collaborative Training

The non-collaborative P^{2}MLF training uses only the *Bob*'s training data, but in this case, use DP during training to avoid leaking *Bob*'s training data.

4.8.3.1 DP (DP-SGD)

Figure 4.8 illustrates the non-collaborative training approach utilizing DP. In this scenario, P^{2}MLF uses DP-SGD in the models training based on CNN1D, LSTM, and MLP. This approach assumes that only *Bob* possesses the training data and seeks to ensure the privacy of the data by incorporating differential privacy into the training process.

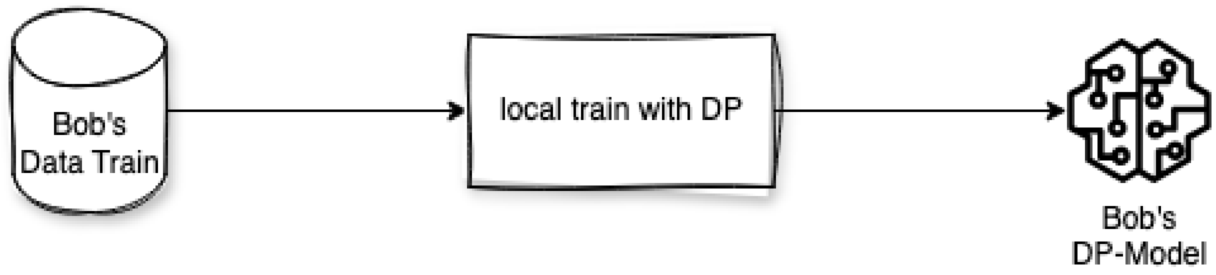

Figure 4.8: P^2MLF Non-Collaborative Training with DP

4.9 Security and Privacy

In this section, security considerations will be made for the preprocessing, model, inference, and training phases when the protocols MPC and DP are involved to ensure input privacy and output privacy. To simplify the explanation, the DGA application (see Chapter 6), which utilizes all phases of the P^2MLF, will be used as an example.

In P^2MLF, collaborative training methods do not prevent adversarial attacks. The principle is to eliminate an attacker who has modified his parameter and is trying to influence the global model. An adversarial attack, for example, could try to influence the global model to, for example, try to change a class considered malignant to benign.

One way to prevent adversarial attacks is to use methods to check whether the parameters sent by a given local model are outliers [120]. In this way, the aggregator would exclude a poisoned local model, which would return the global model without the poisoned model.

A secure aggregator [111] could be built with or without MPC protocols or Homomorphic Encryption (HE).

The following is the list of collaborative training methods in ascending order of privacy guarantees P^2MLF training DP with FL $>$ P^2MLF training FL with DP and aggregator using MPC $>$ P^2MLF training MPC with DP. However, reverses this order when considering the scalability of each method:

1. P^2MLF training DP with FL: Clients perform the training of the local models with DP guarantee and send the parameters to an aggregator that will aggregate the parameters of the local models.

2. P^2MLF training FL with DP and aggregator using MPC: Clients perform training on local models with DP guarantee and send the parameters to a MPC protocol for a secure aggregator that will aggregate the parameters of the local models.

3. P^{2}MLF training MPC with DP: MPC protocols for training ML models with DP guarantees.

4.9.1 Input privacy

The underlying MPC protocols from MP-SPDZ [19] that this work uses in the solutions (replicated secret sharing in the case of 3 computing servers with an honest majority and the semi2k protocol in the case of 2 computing servers without an honest majority) implement a secure arithmetic black-box MPC. They only perform operations over secret shares and no information leaks during the computation over the secret shares. Moreover, this work uses the MPC sub-protocols π_{SIGMOID} for the sigmoid function, π_{SOFTMAX} for the softmax function, π_{MUL} for the secure dot product, π_{TANH} for the hyperbolic tangent, π_{RELU} for relu function, π_{DENSE} for dense layer, and π_{LR} for LR model from MP-SPDZ [19]. All these sub-protocols do not leak any information and are universally composable, Universal Composability (UC), secure. The novel protocols that it proposes ($\pi_{\mathsf{EMBEDDING}}$, π_{CNN1D}, and π_{LSTM}), therefore, do not leak any information to the computing servers responsible for the MPC operations over the secret shares about *Alice's* or *Bob's* inputs; nor any private information leaks to *Alice* or *Bob* other the result of the classification (which *Alice* can reconstruct by getting all shares of the output from the computing server). The protocols UC-securely implement the ideal functionality $\mathcal{F}_{PPCDGA}$ for privacy-preserving classification of domains as DGA or non-DGA that is described below for the case of classification using MLP, CNN1D, or LSTM models.

Therefore, P^{2}MLF follow UC framework to the protocols π_{CNN1D}, π_{DP}, π_{DENSE}, $\pi_{\mathsf{EMBEDDING}}$, $\pi_{\mathsf{LRINFERENCE}}$,$\pi_{\mathsf{LRTRAINING}}$, π_{LSTM}, π_{MUL}, π_{RELU}, π_{SIGMOID}, π_{SOFTMAX}, π_{SUM}, π_{TANH}, $\pi_{\mathsf{FLAGGREGATOR}}$.

Functionality $\mathcal{F}_{PPCDGA}$

$\mathcal{F}_{PPCDGA}$ waits until it receives as input from *Alice* her domain name x and as input from *Bob* his model m for the DGA classification that he trained with DP guarantees.
Upon receiving both inputs, $\mathcal{F}_{PPCDGA}$ locally computes the result y of classification x using the model m and sends y as a public delayed output to *Alice*.

4.9.2 Output privacy

The guarantee that *Alice* does not learn information about individual entries used in the training of *Bob's* model - is provided by the DP guarantees of DP-SGD [40] as utilized by *Bob*. With output privacy, attacks such as membership inference [21] or model inversion [110] can be avoided.

Chapter 5

P^{2}MLF Training Applied in CIDS

This chapter presents collaborative and secure training proposals based on Privacy-Preserving Machine Learning Framework (P^{2}MLF) applying in the Collaborative Intrusion Detection Systems (CIDS) problem. It discusses each proposed method's privacy, security, and performance levels.

A CIDS consists of network data from different parts [106, 121]. This work analyzes the privacy requirements of both input and output data to underscore the necessity of adopting P^{2}MLF. A CIDS is helpful because it enables the creation of a model to detect a wide range of network attacks. Privacy is essential when parties(organizations, devices, or people) must perform model training with sensitive data due to regulatory data privacy requirements.

Thus, secure training will enable parties to collaborate to build a model that takes advantage of each data set's unique characteristics while prioritizing each party's data protection.

The intrusion detection discussed will focus on protecting network data from Internet of Things (IoT) devices. Differential Privacy (DP) is used to perform Logistic Regression (LR) model training, ensuring the privacy of the results and preventing potential leaks in trained models. Input data privacy is on guarantees through Secure Multi-Party Computation (MPC) and Federated Learning (FL), which distribute the training process between parties.

This chapter will not address the prevention of adversarial attacks, where a participant could introduce a compromised model to alter the behavior of the global Machine Learning (ML) model, potentially causing the model to classify a malicious class as benign.

5.1 Introduction

Intrusion Detection System (IDS) detects irregularities and potential threats in network traffic by analyzing incoming data, inspecting packets or flow types, and identifying malicious traffic [122] [123]. Confirmed incidents generate reports that are kept confidential for privacy reasons. However, significant volumes of both benign and malicious traffic information are required to build effective ML models that can learn to detect threats. Organizations can collaborate to expand their incident base and conduct collaborative training. However, many companies need to avoid such collaboration to avoid exposing data that could reveal sensitive customer information [124].

A traditional approach to training ML involves sending data to a trusted third party to consolidate it and use it as training data. However, centralizing data also poses significant privacy and security risks. A centralized approach becomes a single point of failure; if the data center holding the data is compromised, it allows an attacker to leak that information. This vulnerability highlights the need for stringent security measures and robust data protection protocols to protect provisional information from malicious actors [69, 97, 101].

The availability of multiple datasets from different parties allows for developing models (CIDS) capable of identifying a broader range of attacks. However, there are situations where companies holding datasets are willing to contribute their data as long as data privacy guarantees are in place [69, 97, 101].

Information exchange on cyberattack incidents can facilitate rapid decision-making and the implementation of robust countermeasures. Specifically, integrating proactive cyber threat incident sharing with defensive mitigation strategies can increase the resilience of entities to build a collective defense against new or potentially unknown attacks and malware. Therefore, it is crucial to develop platforms, tools, frameworks, and methodologies that enable efficient access and sharing of information on threat events to respond quickly and mitigate potential damages [121].

Considering all the points presented, the main contribution of this work is a CIDS approach that prioritizes the privacy of the parties' network data and the intellectual property of the CIDS model owner. This work uses DP in the LR model training, ensuring output privacy and avoiding potential data leaks in the trained models. This work ensures input privacy by using MPC and FL to distribute training across parties.

5.2 Problem

Consider that two or more parties wish to develop a CIDS utilizing data from all involved parties. However, privacy requirements dictate that only the data owner should have exclusive access to their information. This work will refer to this problem as the problem of training with data privacy.

Figure 5.1 illustrates the first problem discussed in this work. Consider three parties(organizations) with data confidential: *Bob*, *Clara*, and *Dan*. Consider that these parties have confidential

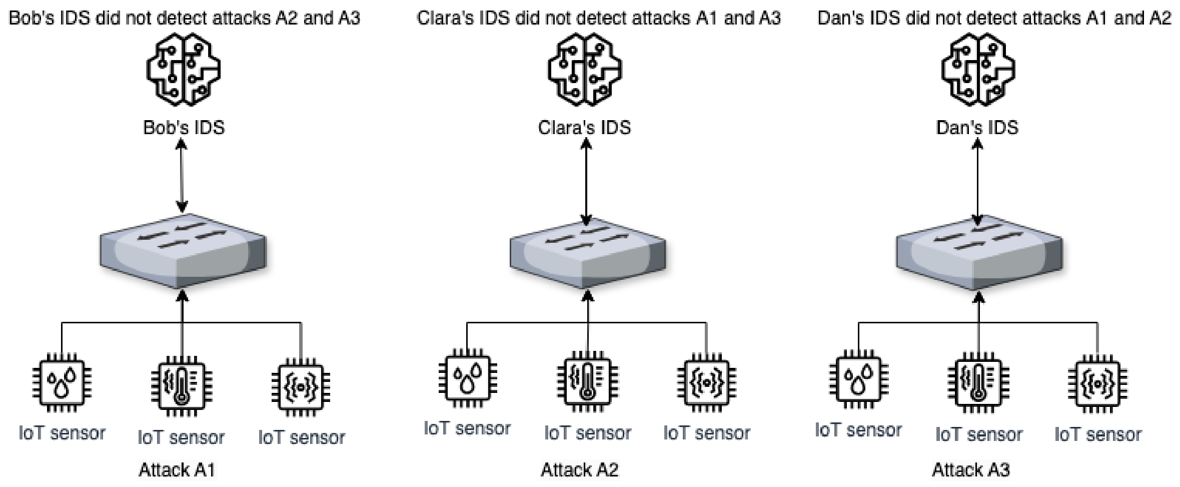

Figure 5.1: Illustration of benefits and challenges for training a CIDS with data from different parties

data traffic on their IoT networks and intend to use these data to create a CIDS. As a result, *Bob*, *Clara*, and *Dan* are seeking a solution that facilitates the training of a CIDS while simultaneously ensuring the privacy of each party's data. They aim to develop a CIDS that upholds training data privacy. By combining the data from the three parties, this work can create a model for a CIDS that efficiently detects attacks *A1*, *A2*, and *A3*. The challenge lies in training the CIDS's ML model with data from the three parties without using a centralized repository or trusting these data to a third trusted party. In other words, one must conduct the process in a way that allows only each data owner to access their respective data.

This problem to be solved is how to perform a CIDS's model training with data from different parties, ensuring that only the data owner can access their respective information.

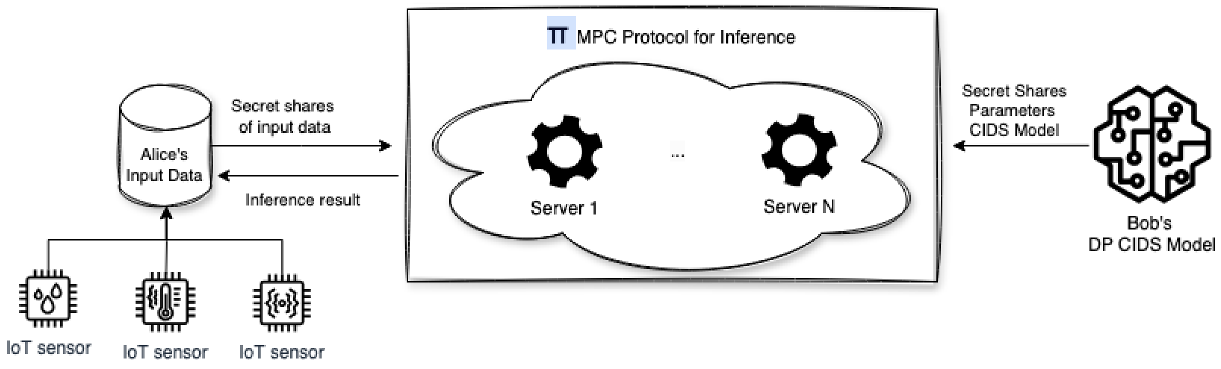

Figure 5.2: Illustration of benefits and challenges of secure inference

Consider a second problem where there are two parties, and one owns the data and wants to perform inference to detect intrusion using a model CIDS. However, there is a privacy requirement on both sides; only the owner must know the data and only the model owner must know the CIDS model.

The second problem, illustrated in Figure 5.1, explains the secure inference. Consider a party called *Alice*, which has sensitive data traffic from the IoT networks. Consider another party, *Bob*, who owns a model for CIDS. Therefore, *Alice* wants *Bob* to provide data inference, ensuring *Alice*'s data privacy; this way, *Bob* will not know *Alice*'s plain text data. The meaning of *Bob*'s privacy guarantee will ensure that *Alice* does not receive any information about the weights of the model trained by *Bob*.

When applied in ML, technologies like MPC, Homomorphic Encryption (HE), and FL are instrumental in safeguarding input privacy and effectively ensuring the confidentiality of a party's data; however, it is crucial to recognize that MPC, HE, and FL, while securing data privacy for parties during model training ML, do not prevent information leakage after completing model training; in essence, these methods do not secure output privacy [41, 110, 125, 126].

This work aims to provide comprehensive privacy protection, addressing input and output privacy challenges in a model CIDS. This framework encompasses three strategies for secure training: integrating MPC with DP, FL with DP, and FL with DP using a protocol MPC to aggregation. Moreover, this framework utilizes MPC to ensure data privacy and security for secure inference.

The challenge in the proposed problem is that existing techniques add a sizeable computational overhead in processing, memory, energy, and network communication. Therefore, this work needs to reconcile privacy techniques and optimize ML models to make the use of a ML model for CIDS viable and with privacy guarantees.

5.3 Motivation to the Problem

The problems discussed in the section 5.2 are justified when parties desire the training sensitive data. In principle, solving the problem of secure ML collaborative training will allow us to create models by aggregating data from different parties and protecting the privacy of the data of each participant in the model and considering that a given model trained with a set of data from various participants will allow us to create a CIDS that can cover attacks that training with isolated data would not be able to.

Solving the problem regarding secure inference protects the owners of sensitive data in scenarios with a malicious service or CIDS tool. Therefore, it protects data from conditions where there is a malicious tool CIDS that can expose the analyzed network data. The solution to the problem also protects the intellectual property of the CIDS model in case of a compromise of the infrastructure that hosts the service that hosts the CIDS.

5.4 Contributions

The following are the contributions outlined in this chapter:

- P^2MLF secure collaborative training methods applied in CIDS problem using private IoT data from different parties, ensuring CIDS models that can detect a wide range of attacks obtained from knowledge of different datasets.
- Input privacy by ensuring that the data is known only to the owner of the information through the use of MPC and FL.
- Output privacy, ensuring that there is no leakage of data used in training models CIDS through DP.
- This proposal guarantees output privacy, ensuring that there is no leakage of data used in training models CIDS through DP.
- Method with protocol MPC for secure aggregation in FL with local model trained with DP.
- Considering *Alice* as the owner of the data and *Bob* as the model CIDS, this work proposes that no information from *Alice*'s data will leak to *Bob*. On the other hand, *Bob* will not need to expose his model, thus guaranteeing the intellectual property of CIDS.
- This solution provides a protocol for CIDS training differentially private using MPC.
- This solution provides a differentially private model FL training instance for CIDS.
- P^2MLF secure inference using MPC protocols with CIDS differentially private [15, 23–25, 34].
- This work provides an empirical performance analysis of the secure training of CIDS differentially private models trained with MPC and FL.
- This work provides an empirical analysis of the different epsilon used in differentially private models and considers the metrics' impact as accuracy, precision, recall, and average F1.

5.5 P^{2}MLF Applied in CIDS

P^{2}MLF guarantees privacy at the input and output of CIDS models. Ensuring privacy at the input maintains the data's privacy as it reaches the model during the training and inference phases. Ensuring privacy at the output prevents data leakage after training the model since even a trained model can potentially leak data from the training process.

This chapter compares the performance of CIDS models to evaluate the efficacy of the suggested framework. These models will be developed using LR and trained with MPC and FL, with adaptations to accommodate training with DP in the LR model.

5.6 CIDS using P^{2}MLF Collaborative Training

Consider n clients who want to collaborate to create a CIDS model. This section will present three possibilities for these clients to collaborate with their data in the training of a CIDS model. The idea is to compare the three collaborative methods regarding runtime and privacy.

5.6.1 P^{2}MLF Collaborative Training DP with FL

In this method, clients use the π_{FLCLIENT} protocol, where each client trains locally with their network data in the clear. The local models use LR with DP guarantees. Afterward, each client sends only the parameters in the clear of the generated local models LR-DP to a server that will run the π_{FLSERVER} protocol.

When running the π_{FLSERVER} protocol, the server receives the parameters in the clear from the models of n clients as input. It calculates the arithmetic mean of each parameter among the n models, returning a global model for each client.

The system achieves the lowest privacy level because it trains the data in the clear and aggregates the models in the clear. On the other hand, the performance is the highest since it does not run MPC protocols.

5.6.2 DP with FL (Aggregation on MPC)

The difference between this method and the 5.6.1 method is that $\pi_{\mathsf{SECUREFLSERVER}}$ uses a MPC protocol to aggregate models.

In this method, clients use the $\pi_{\mathsf{SECUREFLCLIENT}}$ protocol, where each client trains a local LR model with DP based on its network data in the clear, performing training for just one epoch. Each client sends the secret shared parameters of its local model LR-DP to a server running the $\pi_{\mathsf{SECUREFLSERVER}}$ protocol.

When executing the $\pi_{\mathsf{SECUREFLSERVER}}$ protocol, the server receives the secret shared parameters from the models of n clients and computes the protocol MPC to realize the secure aggregation ($\pi_{\mathsf{FLAGGREGATOR}}$) of each parameter across the n models. In the final $\pi_{\mathsf{FLAGGREGATOR}}$, the protocol $\pi_{\mathsf{SECUREFLSERVER}}$ returns a global model with the sum of each parameter for all n clients.

Upon receiving the global model, each client will divide it by n to calculate the average of each of the parameters of the global model.

The system provides an intermediate privacy level because it trains the data in the clear. Still, the aggregation of the models uses the MPC protocol that will compute secret shares of the models. The runtime is shorter than the method with FL and DP.

5.6.3 P^{2}MLF training MPC with DP

The MPC protocol $\pi_{\mathsf{LRTRAINING}}$ is to perform LR training on models with DP guarantees. P^{2}MLF training employs the π_{DP} protocol to generate DP Laplacian noise in MPC, described in [5].

The level of privacy in this case is higher since the training is on the secret shares of the data. On the other hand, the execution time is longer since there are rounds of MPC communication, and there is also more data to compute in secret shares.

5.7 P^{2}MLF Inference of CIDS

In the inference stage, *Alice* must determine whether an instance (network data) is malicious or non-malicious.

Alice shares her network data, *Bob* shares the model parameters with the computing servers, and *Bob*'s model architecture is public knowledge.

The computing servers execute MPC protocols that output *Alice*'s secret shares of the classification result. This proposed framework can perform inference using models with LR architecture.

Alice receives the classification output of her network data in this proposed solution for secure inference. In the binary case, it is malicious or non-malicious.

This output should preserve the privacy of the individual entries of *Bob*'s training data set. This work employs the well-known DP technique to mitigate privacy concerns and provide DP guarantees for *Bob*'s training data set according to Definition 1. This work explicitly trains the LR model with DP guarantees.

In this problem, the P^{2}MLF inference uses MPC protocols π_{SIGMOID} for the sigmoid function, $\pi_{\mathsf{DOTPRODUCT}}$ for the secure dot product, π_{SUM} for the sum, $\pi_{\mathsf{LRINFERENCE}}$ for inference LR, π_{LR} for the training LR. See [19] for a detailed description of these MPC primitives.

5.8 Dataset

The IoT attack dataset CICIoT2023 [1] [99] contains 33 attacks organized in the categories: DDoS, DoS, Recon, Web-based, Brute Force, Spoofing and Mirai [99]. Finally, malicious IoT devices execute all attacks, targeting other IoT devices.

The 39 features of the dataset CICIoT2023 [99] are Header_Length, Protocol Type, Time_To_Live, Rate, fin_flag_number, syn_flag_number, rst_flag_number, psh_flag_number, ack_flag_number, ece_flag_number, cwr_flag_number, ack_count, syn_count, fin_count, rst_count, HTTP, HTTPS, DNS, Telnet, SMTP, SSH, IRC, TCP, UDP, DHCP, ARP, ICMP, IGMP, IPv, LLC, Tot sum, Min, Max, AVG, Std, Tot size, IAT, Number, Variance [99].

There are three binary datasets representing *Bob*, *Clara*, and *Dan*, each containing 6000 examples. *Bob*'s dataset includes 500 examples of each of the following attacks, in addition to 500 examples of benign data: Backdoor Malware, Browser Hijacking, Command Injection, DDoS-ACK Fragmentation, DDoS-HTTP Flood, DDoS-ICMP Flood, DDoS-ICMP Fragmentation, DDoS-PSHACK Flood, DDoS-RSTFIN Flood, DDoS-SYN Flood, and DDoS-SlowLoris.

The dataset of *Clara* has a similar data division to that of *Bob*. It has the following attacks: DDoS SynonymousIP Flood, DDoS TCP Flood, DDoS UDP Flood, DDoS UDP Fragmentation, DNS Spoofing, Dictionary Brute Force, DoS HTTP Flood, DoS SYN Flood, DoS TCP Flood, DoS UDP Flood, MITM Arp Spoofing.

The dataset of *Dan* has a data division similar to that of *Bob*, and *Clara* and has the following attacks: Mirai greeth flood, Mirai greip flood, Mirai udp plain, Recon Host Discovery, Recon OS Scan, Recon Ping Sweep, Recon Port Scan, SQL Injection, Uploading Attack, Vulnerability Scan, XSS.

All data was split and shuffled, with 80% for training data and 20% for testing data for the architecture with binary models.

5.9 Utility-Privacy Trade-Off

Table 5.2 presents an in-depth examination of the balance between utility, as gauged by the model's accuracy, and privacy, as denoted by the degree of DP within a given privacy budget, symbolized as ϵ.

[1]https://www.unb.ca/cic/datasets/iotdataset-2023.html

The privacy budget ϵ serves as a quantifiable indicator of privacy. Generally, a value of ϵ below one signifies a high level of privacy, ranging from one to two, which denotes moderate privacy, and a value of ϵ exceeding two indicates a low level of privacy. An infinity value ϵ corresponds to a model devoid of DP guarantees. This work can verify, as anticipated, that there is an inverse relationship between the model's accuracy and the privacy budget; as the budget diminishes, the model's accuracy decreases, thus propelling it into a higher privacy domain. This phenomenon underscores the inherent trade-off between utility and privacy when implementing DP.

5.10 Security and Privacy

When applied to training CIDS models, this framework's security arises from using MPC protocols (for three and two parties) and carrying out operations on secret shares. This framework guarantees input privacy, meaning that the servers involved in MPC computation learn nothing about *Alice* and *Bob*'s input.

This work also proposes training a CIDS with FL that guarantees input privacy of the parties' data.

This solution ensures privacy in the output through DP [40], thus preventing *Alice* from learning about the individual inputs used in *Bob*'s CIDS model.

This work presents a secure inference using MPC. This solution allows a trained model to be loaded using secure training comprising MPC and FL solutions. This proposal can also load a trained model outside the secure training framework.

The proposed framework also allows for a scenario with a lower security level but higher performance, providing training models with privacy in a shorter execution time.

5.11 Runtimes

This work implemented the MPC-based protocols in three computing servers (3PC) connected over a Gigabit Ethernet local area network. This work used the following underlying MPC protocols available on MP-SPDZ [19] using replicated secret sharing for 3PC. The runtimes are available in Table 5.1 and include communication and computation delays. In table 5.1, the results of the protocol $\pi_{\mathsf{LRINFERENCE}}$ are only for one inference. For the LR model, this work evaluated metrics across various values of ϵ (see Table 5.2). A smaller ϵ corresponds to a higher level of applied noise, while an infinite ϵ represents the absence of noise. The results show that noise significantly impacts the data in this experiment, causing the metrics to decline noticeably.

This work notes that protecting the model with DP does not affect runtimes.

This work used AMD EPYC instances with 32 cores @ 2.9 GHz and 65GB of RAM for the inference and training experiments.

The method using DP with MPC presents a higher level of privacy because it computes data over secret shares, but the runtime to perform the training of this method is longer. On the other hand, the other two methods using FL and with and without $\pi_{\mathsf{FLAGGREGATOR}}$ have runtimes very close. This runtime behavior is because the MPC protocol for $\pi_{\mathsf{FLAGGREGATOR}}$ aggregation receives secret shares from the local models and returns the global model as a sum of all parameters from all involved parties. The average is calculated in each party to reduce the runtime of the $\pi_{\mathsf{FLAGGREGATOR}}$ protocol.

Model	**Runtime seconds**
DP with FL	42.29
DP with FL(Aggregation on MPC)	42.30
DP with MPC	62.9461
$\pi_{\mathsf{LRINFERENCE}}$	0.0076

Table 5.1: P^{2}MLF Runtimes

Noise Epsilon	**Accuracy**	**Precision**	**Recall**	**Average F1**
0.1	0.39	0.85	0.39	0.49
0.5	0.52	0.85	0.52	0.62
1	0.57	0.85	0.57	0.66
∞	0.92	0.85	0.92	0.88

Table 5.2: Metrics for Different Noise Levels

Protocol	**Setting**	**Online Phase**			**Offline Phase**		
		Rounds	**Data Sent(MB)**	**Runtime**	**Rounds**	**Data Sent(MB)**	**Runtime**
$\pi_{\mathsf{LRTRAINING}}$	3PC	142676	1160.13	35.5968	87408	1756.9	25.1032
$\pi_{\mathsf{LRINFERENCE}}$	3PC	14	0.00188	0.002	8	0.043	0.002
$\pi_{\mathsf{FLAGGREGATOR}}$	3PC	2	0.00032	0.0005	-	-	0.00003

Table 5.3: P^{2}MLF MPC Protocols

5.12 Final considerations

This work introduced P^{2}MLF developed for the training and inference of CIDS models, considering privacy preserving at both the input and the output. In other words, this work presented the first DP training to avoid leaky data in CIDS and the first secure inference with MPC and FL. Input privacy ensures that the computational servers, the model owner, or *Bob* cannot learn anything about the classified data. It also ensures that the model's details remain undisclosed to the data owner, *Alice*, and the computational servers. Conversely, to safeguard the data used in training, output privacy implementation prevents extracting individual data entries from the training dataset of the ML model employed within the framework.

This work evaluated the efficacy of secure inference within this framework, which employs LR models trained to use DP and secured by implementing various MPC protocols. Furthermore, this work evaluated the performance of secure training with three methods of this framework: DP with FL, DP with FL(Aggregation on MPC), and DP with MPC.

All three methods guarantee privacy at both input and output but differ in scalability. The method that combines DP and FL presents greater scalability since it allows the aggregator to run on a single server. However, this aggregator becomes a single point of failure since it has direct access to each participant's local models in the clear.

On the other hand, the method that uses DP, FL, and $\pi_{\mathsf{FLAGGREGATOR}}$ offers a better balance between privacy and scalability, being 1.5 faster than the method with MPC and DP. It guarantees privacy at both input and output while maintaining the second-lowest execution time. $\pi_{\mathsf{FLAGGREGATOR}}$ adds an extra level of privacy by generating the global model based on secret shares of the local models, preventing direct access to the participants' model. A possible improvement would be to use $\pi_{\mathsf{FLAGGREGATOR}}$ to detect adversarial attacks through MPC protocols.

The method that combines DP and MPC achieves the highest level of privacy, as it trains a LR model using a MPC protocol and, in the end, adds DP by applying Laplacian noise to the model parameters through a MPC protocol.

Chapter 6

P^{2}MLF Inference Applied in DGA

This chapter proposes a secure inference and non-collaborative secure training strategy based on Privacy-Preserving Machine Learning Framework (P^{2}MLF) applied to detect Domain Generation Algorithms (DGA) and evaluates each proposed method's privacy, security, and performance levels. For nomenclature reasons, the P^{2}MLF inference method will be called secure inference.

This chapter uses the framework P^{2}MLF that combines the benefits of automated and outsourced DGA detection while preserving the data privacy of enterprise network users and DGA detection service providers. This framework does not expose the Domain Name System (DNS) in clear text to DGA detection service providers, nor is the model shared with enterprise network administrators (The parties do not receive any private information). This chapter incorporates techniques that prevent attackers from obtaining additional information about training data or reconstructing the model from the classification results sent to enterprise network administrators.

6.1 Introduction

The applicability of Machine Learning (ML) models to develop classifiers that identify and differentiate benign domains from malicious domains generated by Malicious Software (Malware) based on DGA represents a viable approach, as discussed in [66]. These classifiers automate the detection of Malware in corporate networks.

Advanced deep learning models achieve high accuracy but require large amounts of data for training [64]. Third-party organizations often offer these models through service providers as part of a DGA detection service to meet this demand. In this architecture, the company's DNS traffic goes to the provider, which classifies the domains as malicious or benign and returns the results to the company [65].

However, this outsourced detection architecture raises significant privacy concerns. The company's DNS traffic may contain sensitive information, compromising users' privacy on the corporate network, which raises concerns in the DGA-as-a-service detection paradigm.

An alternative involves providing the ML model directly to the company's network administrators, allowing for local deployment. However, this brings new challenges since the service provider owns the model. Furthermore, the data used in the training of these models may be private, and providing the model to the enterprise exposes these data to attacks, as discussed in [125, 126].

Protecting sensitive data from the enterprise and service providers is essential, but Privacy-Enhancing Technologies (PET) significantly increases computational costs.

This chapter explores how to ensure privacy for both the owner of the ML model and the party performing the DNS domain classification. This chapter shows an end-to-end privacy-preserving architecture for outsourced DGA classification that protects enterprise users and service providers.

This solution allows network administrators to outsource traffic analysis (in this case, DNS traffic) to external parties without risking information leakage. Furthermore, the approach also protects the intellectual property of the ML model by preventing the consuming party from obtaining any information about the model used.

6.2 Problem

This work recalls that *Alice* holds a DNS domain to be classified, and *Bob* holds a ML model that classifies DNS domains as malicious or benign. This framework consists of a set of m untrusted computing servers (Secure Multi-Party Computation (MPC) servers) $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. Although the proposed protocols MPC are general and work for any number of servers, this work demonstrates the protocols proposed for $m = 2$ and $m = 3$. This work assumes pairwise authenticated and private communication channels between the servers. The communication between *Alice*, *Bob*, and any server $S_i \in \mathcal{S}$ is also authenticated and private. The proposed secure classification works as follows:

- Initially, *Alice* and *Bob* convert their private inputs with real values (domains in the case of *Alice* and the model parameters in the case of *Bob*) into fixed-point representations. They then secret-share their respective fixed-point inputs with the computing servers.
- The computing servers then engage in MPC-based communications and computations to execute the MPC protocols to classify *Alice's* domain names using *Bob's* model.
- The inference uses secrets shared among the computing servers at the end of the MPC protocols. The computing servers send their shares to *Alice*. Finally, *Alice* aggregates the secret shares and retrieves the classification result.

6.3 Contributions

The following are the contributions outlined in this chapter:

- Proposition of a new framework for private classification of glsDGA/non-DGA domains, with guarantees input privacy MPC and output privacy Differential Privacy (DP).
- Presentation of the first approach that considers differentially private training of models for DGA classification.
- The solution works with classifiers based on MultiLayer Perceptron (MLP), One-Dimensional Convolutional Neural Network (CNN1D), and Long Short-Term Memory (LSTM). The MPC protocol for LSTM is novel, efficient, and the first on the MPC MP-SPDZ framework [19].
- Evaluation of the proposed framework on real datasets — DGArchive and Alexa — for binary and multiclass classification tasks of domain names. The binary classification problem distinguishes domains as benign or malicious, while the multiclass classification problem identifies the DGA family associated with the malicious domain.
- Empirical analysis of the privacy-utility trade-offs of the approach using MLP, CNN1D and LSTM. It observes that ensuring output privacy slightly degrades accuracy due to noise introduced to ensure DP while using MPC to ensure input privacy does not affect the usefulness of the classification model. The fastest model in terms of inference time is MLP, with a time of 0.07 seconds. The model with the highest accuracy is CNN1D, achieving the accuracy 93% for an epsilon of 5.
- Demonstration of the efficiency of the proposed solution in terms of execution time with two or three computation parts.
- Significant improvements in the performance of MPC protocols were observed with quantization, resulting in a 23% to 42% reduction in inference runtime, without compromising accuracy, in the 3PC setting (using replicated secret sharing). The proposed solution enables near-real-time secure detection of DGA domains.

6.4 P^{2}MLF Applied in DGA Detection

Figure 6.1 presents a high-level proposal for secure DGA inference. In the scheme, *Alice* represents the company, which owns the domains and DNS traffic, while *Bob* represents the service provider, which owns the weights (parameters) of the already trained model ML.

Bob can choose to perform an MLP training, a CNN1D, or an LSTM model, and the selected model must be trained with DP guarantees, as described by [39] to ensure privacy in the output. In addition, *Alice* wants her DNS traffic to be classified by *Bob*. In the

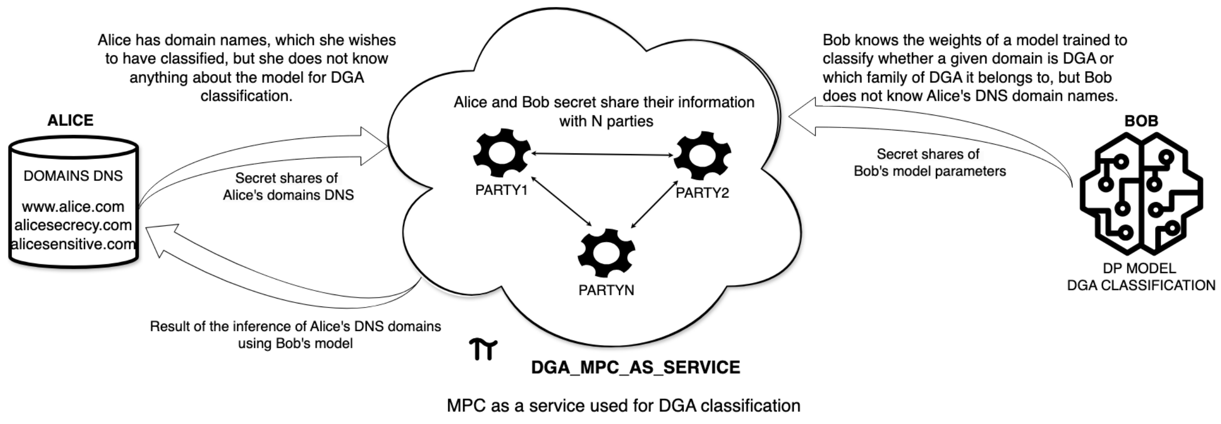
Alice has domain names, which she wishes to have classified, but she does not know anything about the model for DGA classification.
ALICE
DOMAINS DNS
www.alice.com
alicesecrecy.com
alicesensitive.com
Secret shares of Alice's domains DNS
Result of the inference of Alice's DNS domains using Bob's model
Alice and Bob secret share their information with N parties
PARTY1
PARTY2
PARTYN
DGA_MPC_AS_SERVICE
MPC as a service used for DGA classification
Bob knows the weights of a model trained to classify whether a given domain is DGA or which family of DGA it belongs to, but Bob does not know Alice's DNS domain names.
Secret shares of Bob's model parameters
BOB
DP MODEL
DGA CLASSIFICATION

Figure 6.1: The flowchart uses an end-to-end privacy approach to describe the privacy-preserving service for DGA inference. The service provider (*Bob*) trains the DNS domain name classifier with DP-SGD, ensuring DP (output privacy). *Alice* submits new domain names, and *Bob*'s model classifies these domains using MPC. MPC servers execute protocols over encrypted data, maintaining input privacy.

context of this chapter, *Alice*'s DNS data and *Bob*'s model weights are considered private data.

The proposed framework uses MPC techniques [15] to preserve input privacy. To do so, *Alice* and *Bob* secretly share their private data with a set of untrusted computing servers (parties). These MPC servers perform computations on the secret shares, labeling the domain names as benign or malicious, ensuring that:

(P1) None of the parties, including the servers, has direct access to *Alice*'s data or *Bob*'s model parameters.
(P2) Input and output privacy are preserved throughout the classification process.

This approach ensures that the data privacy of both parties is maintained, even in an outsourced computing environment.

(P1) No individual MPC server can obtain any information about the domain names owned by *Alice*;
(P2) No individual MPC server can access any information about the weights of *Bob*'s ML model;
(P3) Only *Alice* should receive the classification result;
(P4) The classification result cannot reveal to *Alice* any private information about the individual inputs used in *Bob*'s training dataset;

This chapter observes that (P4) provides *output privacy* and is achieved when *Bob* trains the model with DP guarantees. Items (P1)–(P3) guarantee *input privacy* that achieves through the novel MPC protocols proposed for inference with a neural network model trained for DGA detection.

MPC protocols typically incur high communication and computation costs, which impact inference runtime and overall performance. To improve the performance of the proposed MPC protocols for secure DGA domain classification, take advantage of the quantization schemes available in TensorFlow (TFLite). The quantization method reduces the precision of the parameters of an ML model, which is typically 32-bit floating point numbers [1]. The method proposes that *Bob* use post-training quantization techniques on the DP-trained model before using it for classification.

6.5 DGA Classifiers using P^2MLF Non-Collaborative Training

In this proposed solution, *Alice* receives the output of her domain classification, which can be a DGA or non-DGA label in the binary case. In the multiclass case, the output indicates a non-DGA label or a specific DGA family.

This method ensures that this output preserves the privacy of the individual entries of *Bob*'s training dataset. To alleviate privacy concerns and provide DP guarantees over *Bob*'s training dataset, is used the technique known as Differentially-Private Stochastic Gradient Descent (DP-SGD) [40], as defined in Definition 1. This method explicitly trains the MLP, CNN1D, and LSTM models with DP guarantees.

This method applies the post-training quantization technique to improve the inference performance of the trained model. This technique reduces the precision of the model weights and activations from 32-bit floating point to lower precision representations. In this method is quantized the model parameters to float16 [2], which results in minimal accuracy loss and allows for more efficient deployment on various hardware, such as GPU and CPU with float32 or float16 support.

The DP guarantees remain in the post-trained quantized model due to the post-processing property of DP. Furthermore, it is possible to adapt this framework to other quantization schemes [34, 127, 128]. Combining DP with quantization techniques can preserve the privacy of *Bob*'s training dataset while accelerating the inference process.

[1]https://www.tensorflow.org/lite/performance/post_training_quantization/

[2]https://www.tensorflow.org/lite/performance/post_training_quantization#float16_quantization

6.6 P^{2}MLF inference of DGA domains

During the inference phase, *Alice* receives an instance (raw text input, such as the domain name string) and calls the inference service to classify it as either a DGA or a non-DGA domain. The process starts by adjusting the length of the given input text to a publicly known value, l, by truncating or padding the input text with zeros.

After this adjustment, the preprocessing encodes the fixed-length text in a one-hot format based on ASCII characters. This results in an array x of dimension $l \times 128$, where the ASCII array contains 128 characters. *Alice* secretly shares the array x, while *Bob* secretly shares the model parameters with the compute servers. The architecture of *Bob*'s model is publicly known.

The computing servers execute the MPC protocols, which produce the secret shares of the classification result for *Alice*. The proposed framework supports inference using models with MLP, CNN1D, and LSTM architectures, all equipped with an embedding layer as the first layer.

Then, the proposed MPC protocols for these models, which perform secure inference, are described.

The implementation of this solution uses MP-SPDZ [19], a public framework to implement various MPC protocols. MP-SPDZ provides a high-level Python interface that describes a circuit in a MPC protocol. It also has circuit representations for ML algorithms such as MLP and 2D Convolutional Neural Network (CNN). However, no implementations are available for LSTM or Embedding Layer (EL) in MP-SPDZ. This work proposes new circuit representations to compute inference in the LSTM and EL networks.

6.7 Dataset

This work obtains the DGA dataset from the DGArchive[3], containing all DGA examples collected until 2019. The method removes DGA families with low representation in the dataset (less than 30K samples), resulting in $1,000,000$ examples from different DGA families, including: bamital, banjori, bedep, beebone, blackhole, bobax, conficker, corebot, cryptolocker, darkshell, dircrypt, dnsbenchmark, dnschanger, downloader, dyre, ekforward, emotet, feodo, fobber, gameover, gameover_p2p.csv, gozi, gspy, hesperbot, locky, madmax, matsnu, modpack, Murofet, Murofetweekly, Necurs, Nymaim, Oderoor, Padcrypt, Proslikefan, Pushdo, Pushdotid, Pykspa, Pykspa2, Pykspa2s, Qadars, Qakbot, Ramdo, Ramnit, Ranbyus, Randomloader, Redyms, Rovnix, Shifu, Simda, Sisron, Suppobox,

[3]https://dgarchive.caad.fkie.fraunhofer.de/welcome/

Sutra, Symmi, Szribi Temped, reve, tinba, torpig, tsifiri, urlzone, vawtrak, virut, volatilitycedar and xxhex.

For non-DGA domains, approximately $1,000,000$ domains were acquired from the latest known version of the "Alexa top 1 million domains" dataset[4] and used in the training of the model to identify legitimate domains.

All data was shuffled and split into 80% for training and 20% for testing across binary and multiclass model architectures.

The preprocessing converts the alphanumeric characters representing the domain names to lowercase for use in the model. Then, it converts each character to the corresponding ASCII code, ranging from 0 to 127. The maximum length of an ASCII domain string is 64 characters. The method is padded with zeros for domains shorter than 64 characters until the desired length.

This work evaluated the experimental results by comparing the CNN1D, LSTM, and MLP models using secure MPC, both for binary and multiclass DGA detection, with and without applying DP.

In addition, after training with DP, the tested model quantization reduced the weights to 16 bits, which provided performance gains during the inference phase with MPC.

6.8 Model Architectures and Parameters

All trained models use an embedding layer as the first layer. The input consists of a vector of 64 numeric elements, resulting from converting each character into its respective ASCII code. After the embedding layer, the result is a matrix with dimensions 128 by 128.

In binary models, the last layer is a dense layer with one neuron, using the sigmoid activation function, the binary cross-entropy loss function, and the Adam optimizer.

In multiclass models, the last layer comprises a dense layer with 65 neurons, representing all DGA families, with the softmax activation function. The loss function used is a sparse categorical cross-entropy, and the optimizer is Adam, with a learning rate of 0.001, batch size of 64, and 30 training epochs.

This work details below the additional layers for each of the architectures used [5]:

- **MLP:** The MLP models, both binary and multiclass, have a flatten layer, which transforms the data resulting from the embedding layer into a one-dimensional representation. Next is a dense layer with 100 neurons, using the ReLU activation function and a dropout rate of 0.1. The binary MLP model has $835,785$ parameters, while the multiclass model has $842,249$ parameters.

[4]`https://en.wikipedia.org/wiki/Alexa_Internet`

[5]`https://github.com/ricardojmmaia/private-dga-detection`

- **CNN1D:** The CNN1D models, binary and multiclass, have (1) a CNN1D layer with 32 filters, the kernel of size 2, ReLU activation function, and a dropout rate of 0.1; (2) a flatten layer that converts the output of the previous layer into a one-dimensional representation; (3) a dense layer with 100 neurons, ReLU activation, and a dropout rate of 0.1. The binary CNN1D model contains $226,409$ parameters, while the multiclass model has $232,671$ parameters.

- **LSTM:** The LSTM models, binary and multiclass, include (1) a LSTM layer with 32 units, ReLU activation, and a dropout rate of 0.1; (2) a flatten layer to transform the output of the previous layer into one-dimensional data; (3) a dense layer with 100 neurons, ReLU activation function, and a dropout rate of 0.1. The binary LSTM model has $48,713$ parameters, while the multiclass model has $55,177$ parameters.

The DP-SGD parameters used in the experiments are available in TensorFlow Privacy [6]. The values are as follows: the delta is 6.188×10^{-7}, the clipping norm is 1, and the number of microbatches is 1. Regarding the noise multipliers, for an epsilon of 0.1, the value is 2.51; for an epsilon of 2, it is 0.61; and for an epsilon of 5, it is 0.46.

6.9 Utility-Privacy Trade-Off

The table 6.1 presents a detailed analysis of the trade-off between utility (measured by model accuracy) and privacy, represented by the privacy budget, denoted by ϵ, of the DP mechanism for all models.

The privacy budget ϵ quantifies the level of privacy guaranteed. In general, values of ϵ less than one indicate high privacy, values between one and two suggest moderate privacy, while values greater than two are considered low privacy. A ϵ equal to infinity indicates a model with no guarantees DP. As expected, the model's accuracy decreases as the privacy budget decreases, reflecting greater privacy protection. The accuracy reduces even more in the multiclass model.

This analysis shows the trade-off between utility and privacy when applying DP. In addition, Table 6.1 shows the accuracy levels of each model with quantization. Quantization did not change the accuracy of the models, although it significantly accelerated performance compared to non-quantized models.

[6]https://www.tensorflow.org/responsible_ai/privacy/api_docs/python/tf_privacy/DPKerasAdamOptimizer

Model	**Non-Quantized, $\epsilon =$**				**Quantized, $\epsilon =$**			
	0.1	2	5	∞	0.1	2	5	∞
CNN1D binary	90%	93%	93%	99%	90%	93%	93%	99%
CNN1D multiclass	25%	47%	53%	88%	25%	47%	53%	88%
LSTM binary	88%	91%	92%	97%	88%	91%	92%	97%
LSTM multiclass	23%	46%	51%	88%	23%	46%	51%	88%
MLP binary	90%	93%	93%	96%	90%	93%	93%	96%
MLP multiclass	24%	46%	51%	87%	24%	46%	51%	87%

Table 6.1: Results related to the accuracy of DGA inference were obtained for different noise levels, both with and without the application of quantization [3].

6.10 Runtimes

This chapter presents MPC-based inference in the scenario of two compute servers (2PC) and three compute servers (3PC) connected by a local gigabit Ethernet network. Inference experiments use three Azure instances with 32 Intel(R) Xeon(R) Platinum 8272CL CPU cores at 2.60 GHz and 64 GB of RAM. The underlying protocols MPC used in the experiments, available in MP-SPDZ [19], are semi2k for 2PC and replicated secret sharing for 3PC.

The runtimes, which include communication and computation delays, are described in Table 6.2. The MLP model showed the best execution times, while CNN1D achieved the highest accuracy, especially for higher values of ϵ (see Table 6.1).

The runtimes do not change with the model using DP guarantees. Furthermore, the corruption threshold significantly impacts execution times. Protocols involving three parties (with an honest majority) tend to be faster than those with two parties, where there is no honest majority.

This chapter also performed inference experiments based on MPC, applying quantization to the models after training with DP. The resulting runtimes are presented in table 6.3.

The quantization decreases the inference runtime by approximately 23% to 42% when the method uses the replicated secret sharing protocol (with three parties). In the 2PC setting (using the semi2k protocol), the reduction was 2% for binary MLP, 4% for multiclass MLP, 1% for CNN1D and 42% for LSTM.

This work uses post-training float16 quantization to optimize models aggressively [128]. By reducing the precision of the model weights to 16 bits, this approach decreases runtimes by approximately 23% to 42% in a 3PC setting (using replicated secret sharing), with minimal impact on accuracy [3].

Model	Setting	Inference Time (sec)	Rounds	Data Sent (MB)
MLP Binary	3PC	0.0778787	2773	17.217
CNN1D Binary	3PC	0.319441	8551	21.6062
LSTM Binary	3PC	10.4153	195131	1485.53
MLP Multiclass	3PC	0.133239	3615	24.8676
CNN1D Multiclass	3PC	0.359278	9382	29.0157
LSTM Multiclass	3PC	10.5449	197123	1489.92
MLP Binary	2PC	14.2051	42923	3951.16
CNN1D Binary	2PC	14.2954	54151	3983.37
LSTM Binary	2PC	103.472	441023	26752.6
MLP Multiclass	2PC	14.577	44599	4054.73
CNN1D Multiclass	2PC	14.6503	55831	4089.34
LSTM Multiclass	2PC	104.077	443895	26820.1

Table 6.2: Inference using MPC protocol [3].

Model	Setting	Inference Time (sec)	Rounds	Data Sent (MB)
MLP Binary	3PC	0.0593449	2469	12.3028
CNN1D Binary	3PC	0.223521	7371	12.3982
LSTM Binary	3PC	5.9959	137044	545.121
MLP Multiclass	3PC	0.076885	3038	15.0529
CNN1D Multiclass	3PC	0.260958	7940	15.1483
LSTM Multiclass	3PC	6.15767	138573	546.706
MLP Binary	2PC	13.813	42723	3845.97
CNN1D Binary	2PC	14.1394	51795	3937.96
LSTM Binary	2PC	60.3208	310659	15352.2
MLP Multiclass	2PC	13.9138	44599	3983.37
CNN1D Multiclass	2PC	14.4177	54151	4054.73
LSTM Multiclass	2PC	60.247	312951	15388.2

Table 6.3: inference using MPC protocol with quantization applied after training with DP [3].

6.11 Final considerations

The multiclass models were more sensitive to the noise added by DP than the binary models. All the evaluated models used an architecture with a first layer of EL and differed in the subsequent layer with the use of three options: MLP, LSTM, or CNN1D.

Considering that the runtimes are for one input example, the models with the LSTM layer presented worse runtime. The models using MLP presented shorter runtimes than CNN1D, but in compensation, the models with CNN1D presented higher accuracy than MLP. If the objective is a model with greater accuracy, the models with CNN1D are an option, but considering the runtime, the models with MLP stand out.

The 16-bit quantization applied to the models after training reduced runtime, rounds, and data sent and did not affect the accuracy of the models. Other techniques can be used to optimize further the presented MPC protocols, which is to generate larger circuits at compile time and thus further reduce communication rounds.

Chapter 7

Conclusions and Future Work

Privacy-Preserving Machine Learning Framework (P^{2}MLF) is introduced as a robust approach to ensuring Privacy-Preserving Machine Learning (PPML), meeting both input and output privacy requirements. The design of Secure Multi-Party Computation (MPC) protocols ensures that information is not leaked and inherits the characteristics of secure Universal Composability (UC) protocols. Designed to be versatile, P^{2}MLF is presented in a framework, making it adaptable for another application that demands privacy in Machine Learning (ML). However, a limitation of P^{2}MLF is its inability to prevent adversarial attacks.

Another limitation of P^{2}MLF is the higher energy consumption due to the requirement of more than two servers in MPC-based protocols. Moreover, Privacy-Enhancing Technologies (PET) technologies introduce significant computational overhead, affecting runtime, memory usage, and the number of communication rounds, especially compared to ML models trained on plaintext data.

This work uses P^{2}MLF to create the first framework that implements outsourced Domain Generation Algorithms (DGA) detection with privacy-preserving guarantees at both input and output levels. Input privacy ensures that no information about the Domain Name System (DNS) domain is leaked to the computation servers or to the model owner (*Bob*), and no data about the model is leaked to the domain owner (*Alice*) or the computation servers. The privacy of the output ensures that the results of the computation do not reveal individual data from the training set of the ML model.

This work also introduces MPC protocols for Long Short-Term Memory (LSTM) and Embedding Layer (EL) in the MP-SPDZ framework, advancing the practical application of ML with privacy and security.

The performance of Convolutional Neural Network (CNN), LSTM, and MultiLayer Perceptron (MLP) trained with Differential Privacy (DP) and inferred with MPC protocols were analyzed and compared. One-Dimensional Convolutional Neural Network (CNN1D) demonstrated the best balance between accuracy and performance in the two-party scenario MPC. At the same time, there is a balance in the performance trade-off between CNN and MLP for three computation servers.

Post-training float16 quantization with DP improved 23% to 42% runtimes without significant accuracy loss when applied in a three-party setup with an honest majority.

This work focuses on performance against honest but curious adversaries. Considering entirely malicious adversaries would likely increase inference times, especially in scenarios with a dishonest majority. Developing efficient protocols for this context is an open challenge.

The proposed solution assumes that the players input the correct data into the protocol. In the case of *Bob*, there is nothing to prevent adversarial or out-of-distribution input from being provided, which represents a potential limitation.

Training a Collaborative Intrusion Detection Systems (CIDS) using P^{2}MLF is one of the advances presented in this work, with the advantage of proposing three distinct approaches for training a ML model using data from different parties. One of the main contributions is the in-depth discussions on the levels of scalability and privacy offered by these three collaborative training approaches available in P^{2}MLF. The method that uses Federated Learning (FL) and a MPC protocol to aggregate local models with DP presents the best balance between scalability and privacy, being 1.50 times faster than the proposal with the highest level of privacy, which is a MPC protocol for training models with DP.

However, a limitation identified in the training proposed for CIDS is the inability to prevent adversarial attacks with privacy guarantees. This limitation is particularly critical because, even with privacy guaranteed in the input and output data, one of the parties may attempt to poison the global model in a collaborative training environment. To cite an example of an adversarial attack, an attacker can manipulate the model to misclassify an intrusion or malware as being in a benign class.

Some possible applications and developments of the methods developed in this thesis include:

- Extend to general natural language classification tasks. Including verifies the possibility of federated training of a Large Language Model, including DP guarantees.
- Build MPC protocols to mitigate adversarial attacks during training with FL.
- Research the impacts of secure aggregator on MPC protocols to prevent adversarial attacks considering local models trained with DP.
- This work must investigate whether DP disproportionately affects underrepresented domains in these datasets, leaving this question for future research.

- In terms of social impact, implementing these solutions can increase energy consumption, considering that more than one server performs the inference instead of a single central server.
- Research techniques to reduce the size of MPC circuits.
- Accelerate the pre-generation of Beaver triples for use in MPC protocols, aiming to reduce runtime and communication rounds. Inference methods that rely on dot product operations, such as Logistic Regression (LR), can apply this optimization. A potential extension of this research involves comparing the preprocessing performance across different hardware platforms, including CPU, GPU, and FPGA.
- Research on synthesizing depth-optimized MPC circuits and increasing circuit width to reduce communication rounds and runtime.
- Research methods to validate whether a model uses DP while preserving the privacy of the data and the ML model itself.
- To evaluate the impact of a MPC protocol that performs model aggregation on FL while mitigating adversarial attacks such as model poisoning and incorporating Laplacian noise into the generated global model.

References

[1] I. Goodfellow *et al.*, "Deep learning-ian goodfellow, yoshua bengio, aaron courville," *Adapt. Comput. Mach. Learn*, 2016. ix, 1, 19

[2] A. Belenguer, J. Navaridas, and J. A. Pascual, "A review of federated learning in intrusion detection systems for iot," *arXiv preprint arXiv:2204.12443*, 2022. ix, 1, 32

[3] R. J. Maia, D. Ray, S. Pentyala, R. Dowsley, M. De Cock, A. C. Nascimento, and R. Jacobi, "An end-to-end framework for private dga detection as a service," *PloS one*, vol. 19, no. 8, p. e0304476, 2024. ix, xix, 1, 2, 22, 23, 24, 25, 29, 40, 72, 73

[4] J. A. de Oliveira, V. P. Gonçalves, R. I. Meneguette, R. T. de Sousa, D. L. Guidoni, J. C. Oliveira, and G. P. Rocha Filho, "F-nids — a network intrusion detection system based on federated learning," *Computer Networks*, vol. 236, p. 110010, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623004553 ix, 1, 31

[5] S. Pentyala, D. Railsback, R. Maia, R. Dowsley, D. Melanson, A. Nascimento, and M. D. Cock, "Training differentially private models with secure multiparty computation," Cryptology ePrint Archive, Paper 2022/146, 2022, https://eprint.iacr.org/2022/146. [Online]. Available: https://eprint.iacr.org/2022/146 ix, 1, 2, 7, 44, 48, 49, 50, 59

[6] A. Agarwal, R. Dowsley, N. D. McKinney, D. Wu, C.-T. Lin, M. De Cock, and A. C. A. Nascimento, "Protecting privacy of users in brain-computer interface applications," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 8, pp. 1546–1555, 2019. ix, 1

[7] E. Debie, N. Moustafa, and M. T. Whitty, "A privacy-preserving generative adversarial network method for securing eeg brain signals," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8. ix, 1

[8] National Institute of Standards and Technology (NIST), "Nist special publication 800-226 (initial public draft): Guidance on evaluating privacy-preserving techniques for artificial intelligence," National Institute of Standards and Technology, Tech. Rep., 2023, initial Public Draft. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-226.ipd.pdf ix, 1

[9] J. Vos, S. Pentyala, S. Golob, R. Maia, D. Kelley, Z. Erkin, M. De Cock, and A. Nascimento, "Privacy-preserving membership queries for federated anomaly

detection," *Proceedings on Privacy Enhancing Technologies*, vol. 3, pp. 186–201, 2024. ix, 1, 2, 6

[10] European Parliament and Council, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation)," 2016, official Journal of the European Union, L119, 1-88. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj x, xi, 3

[11] R. E. Shawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," *CoRR*, vol. abs/1906.02287, 2019. [Online]. Available: http://arxiv.org/abs/1906.02287 x, 2

[12] M. D. Cock, R. Dowsley, C. Horst, R. Katti, A. C. A. Nascimento, S. C. Newman, and W.-S. Poon, "Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation," Cryptology ePrint Archive, Report 2016/736, 2016, https://eprint.iacr.org/2016/736. x, 2

[13] A. Agarwal, R. Dowsley, N. D. McKinney, D. Wu, C. Lin, M. D. Cock, and A. C. A. Nascimento, "Protecting privacy of users in brain-computer interface applications," *CoRR*, vol. abs/1907.01586, 2019. [Online]. Available: http://arxiv.org/abs/1907.01586 x, 2

[14] "The need for privacy with public digital contact tracing during the covid-19 pandemic - the lancet digital health," https://www.thelancet.com/journals/landig/article/PIIS2589-7500(20)30133-3/fulltext, (Accessed on 09/23/2021). x, 2

[15] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing.* Cambridge University Press, 2015. x, 3, 8, 9, 10, 11, 57, 67

[16] "LEI N° 13.709, DE 14 DE AGOSTO DE 2018," 2018. [Online]. Available: https://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm xi

[17] Apple, "Child sexual abuse material (csam) detection - technical summary," https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf, Agosto 2021, (Accessed on 10/17/2021). xi, 4

[18] A. Drichel, M. A. Gurabi, T. Amelung, and U. Meyer, "Towards privacy-preserving classification-as-a-service for dga detection," in *2021 18th International Conference on Privacy, Security and Trust (PST)*, 2021, pp. 1–10. xii, 5, 29, 39

[19] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. [Online]. Available: https://doi.org/10.1145/3372297.3417872 xiii, 5, 10, 11, 33, 34, 38, 39, 41, 44, 49, 52, 60, 61, 66, 69, 72

[20] B. Sowmiya, V. Abhijith, S. Sudersan, R. Sakthi Jaya Sundar, M. Thangavel, and P. Varalakshmi, "A survey on security and privacy issues in contact tracing application of covid-19," *SN computer science*, vol. 2, pp. 1–11, 2021. 1

[21] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, “Membership inference attacks from first principles,” in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1897–1914. 3, 32, 52

[22] D. Melanson, R. Maia, H.-S. Kim, A. Nascimento, and M. De Cock, “Secure multi-party computation for personalized human activity recognition,” *Neural Processing Letters*, pp. 1–27, 2023. 6

[23] D. Evans, V. Kolesnikov, and M. Rosulek, “A pragmatic introduction to secure multi-party computation,” *Found. Trends Priv. Secur.*, vol. 2, no. 2–3, p. 70–246, Dec. 2018. [Online]. Available: https://doi.org/10.1561/3300000019 8, 10, 13, 14, 15, 16, 57

[24] Y. Lindell, “Secure multiparty computation,” *Commun. ACM*, vol. 64, no. 1, p. 86–96, Dec. 2020. [Online]. Available: https://doi.org/10.1145/3387108 8, 10, 57

[25] D. Catalano, R. Cramer, G. Di Crescenzo, I. Darmgård, D. Pointcheval, T. Takagi, R. Cramer, and I. Damgård, *Multiparty computation, an introduction*. Springer, 2005, pp. 41–87. 8, 10, 15, 57

[26] A. C. Yao, “Protocols for secure computations (extended abstract),” in *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, 1982, pp. 160–164. [Online]. Available: https://doi.org/10.1109/SFCS.1982.38 9

[27] ——, “How to generate and exchange secrets (extended abstract),” in *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. IEEE Computer Society, 1986, pp. 162–167. [Online]. Available: https://doi.org/10.1109/SFCS.1986.25 9

[28] S. Micali, O. Goldreich, and A. Wigderson, “How to play any mental game,” in *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*. ACM New York, 1987, pp. 218–229. 9

[29] D. Chaum, I. B. Damgård, and J. van de Graaf, “Multiparty computations ensuring privacy of each party’s input and correctness of the result,” in *Advances in Cryptology — CRYPTO ’87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 87–119. 9

[30] D. Beaver, S. Micali, and P. Rogaway, “The round complexity of secure protocols,” in *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, ser. STOC ’90. New York, NY, USA: Association for Computing Machinery, 1990, p. 503–513. [Online]. Available: https://doi.org/10.1145/100216.100287 9

[31] M. Ben-Or, S. Goldwasser, and A. Wigderson, “Completeness theorems for non-cryptographic fault-tolerant distributed computation,” in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC ’88. New York, NY, USA: Association for Computing Machinery, 1988, p. 1–10. [Online]. Available: https://doi.org/10.1145/62212.62213 9

[32] D. Chaum, C. Crépeau, and I. Damgard, "Multiparty unconditionally secure protocols," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 11–19. [Online]. Available: https://doi.org/10.1145/62212.62214 9

[33] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '89. New York, NY, USA: Association for Computing Machinery, 1989, p. 73–85. [Online]. Available: https://doi.org/10.1145/73007.73014 9

[34] A. Dalskov, D. Escudero, and M. Keller, "Secure evaluation of quantized neural networks," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 4, p. 355–375, Aug 2020. [Online]. Available: http://dx.doi.org/10.2478/popets-2020-0077 10, 57, 68

[35] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proceedings of the 2001 workshop on New security paradigms*, 2001, pp. 13–22. 10

[36] D. Escudero, "An introduction to secret-sharing-based secure multiparty computation," Cryptology ePrint Archive, Paper 2022/062, 2022, https://eprint.iacr.org/2022/062. [Online]. Available: https://eprint.iacr.org/2022/062 10, 11, 12, 16

[37] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology—CRYPTO'91: Proceedings 11*. Springer, 1992, pp. 420–432. 12

[38] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of CRYPTOLOGY*, vol. 13, pp. 143–202, 2000. 16

[39] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, p. 211–407, 2013. 16, 17, 66

[40] 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*, 2016, pp. 308–318. 17, 52, 61, 68

[41] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. 18, 56

[42] Y. J. Wong, M.-L. Tham, B.-H. Kwan, and Y. Owada, "Fedddrl: federated double deep reinforcement learning for heterogeneous iot with adaptive early client termination and local epoch adjustment," *Sensors*, vol. 23, no. 5, p. 2494, 2023. 18

[43] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression.* John Wiley & Sons, 2013, vol. 398. 19

[44] I. Politis, G. Georgiadis, A. Kopsacheilis, A. Nikolaidou, C. Sfyri, and S. Basbas, "A route choice model for the investigation of drivers&rsquo; willingness to choose a flyover motorway in greece," *Sustainability*, vol. 15, no. 5, 2023. [Online]. Available: https://www.mdpi.com/2071-1050/15/5/4614 19

[45] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," 2015. 20

[46] P. Vij, S. Nikam, and A. Bhatia, "Detection of algorithmically generated domain names using lstm," in *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*, 2020, pp. 1–6. 20, 28

[47] S. Akarsh, S. Sriram, P. Poornachandran, V. K. Menon, and K. P. Soman, "Deep learning framework for domain generation algorithms prediction using long short-term memory," in *2019 5th International Conference on Advanced Computing Communication Systems (ICACCS)*, 2019, pp. 666–671. 20, 28

[48] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, "Dga botnet detection using supervised learning methods," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*, ser. SoICT 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 211–218. [Online]. Available: https://doi.org/10.1145/3155133.3155166 20

[49] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A lstm based framework for handling multiclass imbalance in dga botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217317320 20, 28

[50] G. S. Josan and J. Kaur, "Lstm network based malicious domain name detection," *International Journal of Engineering and Advanced Technology (IJEAT) ISSN*, vol. 8, pp. 2249–8958, 2019. 20, 28

[51] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016. 20, 28

[52] Y. Qiao, B. Zhang, W. Zhang, A. K. Sangaiah, and H. Wu, "Dga domain name classification method based on long short-term memory with attention mechanism," *Applied Sciences*, vol. 9, no. 20, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4205 20, 28

[53] P. Lison and V. Mavroeidis, "Automatic detection of malware-generated domains with recurrent neural models," 2017. 20

[54] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting dga domains with recurrent neural networks and side information," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES '19. New York, NY, USA: Association for Computing

Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3339252.3339258 20, 28

[55] C. Olah, “Understanding lstm networks,” https://colah.github.io/posts/2015-08-Understanding-LSTMs/, accessed: 2023-07-04. 20

[56] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” *Advances in neural information processing systems*, vol. 28, 2015. 21

[57] “Enisa threat landscape - the year in review — enisa,” https://www.enisa.europa.eu/publications/year-in-review, 2020, (Accessed on 09/09/2021). [Online]. Available: https://www.enisa.europa.eu/publications/year-in-review/view/++widget++form.widgets.fullReport/@@download/ETL2020+-+A+year+in+review+A4.pdf 23

[58] ENISA, “Enisa etl2020 - malware,” https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends/etl-review-folder/etl-2020-malware, 2020, (Accessed on 09/09/2021). 23

[59] McAfee, “Mcafee labs threats report, april 2021,” https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-apr-2021.pdf, april 2021, (Accessed on 09/09/2021). 23

[60] “Dbir - data breach investigations report,” https://enterprise.verizon.com/resources/reports/2021-data-breach-investigations-report.pdf, 2021, (Accessed on 09/09/2021). 23

[61] C. Patsakis and F. Casino, “Exploiting statistical and structural features for the detection of domain generation algorithms,” *Journal of Information Security and Applications*, vol. 58, p. 102725, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212620308632 23

[62] A. Drichel, U. Meyer, S. Schüppen, and D. Teubert, “Analyzing the real-world applicability of dga classifiers,” in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES ’20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3407023.3407030 24, 29

[63] M. Pereira, S. Coleman, B. Yu, M. DeCock, and A. Nascimento, “Dictionary extraction and detection of algorithmically generated domain names in passive dns traffic,” in *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21.* Springer, 2018, pp. 295–314. 24

[64] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, “Character level based detection of dga domain names,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8. 24, 28, 29, 64

[65] B. Yu, D. L. Gray, J. Pan, M. D. Cock, and A. C. A. Nascimento, “Inline dga detection with deep networks,” in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 683–692. 24, 28, 29, 64

[66] B. Yu, J. Pan, D. Gray, J. Hu, C. Choudhary, A. C. A. Nascimento, and M. De Cock, "Weakly supervised deep learning for the detection of domain generation algorithms," *IEEE Access*, vol. 7, pp. 51 542–51 556, 2019. 24, 28, 64

[67] S. Kumar and A. Bhatia, "Detecting domain generation algorithms to prevent ddos attacks using deep learning," in *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2019, pp. 1–4. 26, 28

[68] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving ddos attack detection using cross-domain traffic in software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 628–643, 2018. 26, 31

[69] E. C. P. Neto, S. Dadkhah, and A. A. Ghorbani, "Collaborative ddos detection in distributed multi-tenant iot using federated learning," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, 2022, pp. 1–10. 26, 54

[70] R. Sharifnya and M. Abadi, "A novel reputation system to detect dga-based botnets," in *ICCKE 2013*, 2013, pp. 417–423. 27

[71] S. Li, T. Huang, Z. Qin, F. Zhang, and Y. Chang, "Domain generation algorithms detection through deep neural network and ensemble," in *Companion Proceedings of The 2019 World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 189–196. [Online]. Available: https://doi.org/10.1145/3308558.3316498 27

[72] L. Sidi, Y. Mirsky, A. Nadler, Y. Elovici, and A. Shabtai, "Helix: Dga domain embeddings for tracking and exploring botnets," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ser. CIKM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2741–2748. [Online]. Available: https://doi.org/10.1145/3340531.3416022 27

[73] M. I. Ashiq, P. Bhowmick, M. S. Hossain, and H. S. Narman, "Domain flux-based dga botnet detection using feedforward neural network," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, 2019, pp. 1–6. 27

[74] J. Mao, J. Zhang, Z. Tang, and Z. Gu, "Dns anti-attack machine learning model for dga domain name detection," *Physical Communication*, vol. 40, p. 101069, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874490719309036 27

[75] J. Huang, P. Wang, T. Zang, Q. Qiang, Y. Wang, and M. Yu, "Detecting domain generation algorithms with convolutional neural language models," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 1360–1367. 27

[76] S. Zhou, L. Lin, J. Yuan, F. Wang, Z. Ling, and J. Cui, "Cnn-based dga detection with high coverage," in *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2019, pp. 62–67. 27

[77] D. S. Berman, “Dga capsnet: 1d application of capsule networks to dga detection,” *Information*, vol. 10, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/5/157 28

[78] C. Chen, L. Pan, and X. Xie, “Dga domain name detection based on bigru-mcnn,” in *Proceedings of the 2019 4th International Conference on Intelligent Information Processing*, ser. ICIIP 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 315–319. [Online]. Available: https://doi.org/10.1145/3378065.3378126 28

[79] H. Shahzad, A. R. Sattar, and J. Skandaraniyam, “Dga domain detection using deep learning,” in *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, 2021, pp. 139–143. 28

[80] Y. Zhang, “Automatic algorithmically generated domain detection with deep learning methods,” in *2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, 2020, pp. 463–469. 28

[81] L. Yang, G. Liu, Y. Dai, J. Wang, and J. Zhai, “Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework,” *IEEE Access*, vol. 8, pp. 82 876–82 889, 2020. 28

[82] S. K, P. Balakrishna, V. Ravi, and S. KP, “Deep learning based frameworks for handling imbalance in dga, email, and url data analysis,” 2020. 28

[83] Z. Liu, Y. Zhang, Y. Chen, X. Fan, and C. Dong, “Detection of algorithmically generated domain names using the recurrent convolutional neural network with spatial pyramid pooling,” *Entropy*, vol. 22, no. 9, 2020. [Online]. Available: https://www.mdpi.com/1099-4300/22/9/1058 28

[84] X. Yun, J. Huang, Y. Wang, T. Zang, Y. Zhou, and Y. Zhang, “Khaos: An adversarial neural network dga with high anti-detection ability,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2225–2240, 2020. 28

[85] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. Pham, S. K. Padannayil, and K. Simran, “A visualized botnet detection system based deep learning for the internet of things networks of smart cities,” *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4436–4456, 2020. 28

[86] Y. Li, K. Xiong, T. Chin, and C. Hu, “A machine learning framework for domain generation algorithm-based malware detection,” *IEEE Access*, vol. 7, pp. 32 765–32 782, 2019. 28

[87] J. J. Koh and B. Rhodes, “Inline detection of domain generation algorithms with context-sensitive word embeddings,” in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2966–2971. 28

[88] A. Cucchiarelli, C. Morbidoni, L. Spalazzi, and M. Baldi, “Algorithmically generated malicious domain names detection based on n-grams features,” *Expert Systems with Applications*, vol. 170, p. 114551, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420311957 28

[89] I. Yilmaz, A. Siraj, and D. Ulybyshev, “Improving dga-based malicious domain classifiers for malware defense with adversarial machine learning,” in *2020 IEEE 4th Conference on Information Communication Technology (CICT)*, 2020, pp. 1–6. 28

[90] R. Sivaguru, J. Peck, F. Olumofin, A. Nascimento, and M. De Cock, “Inline detection of dga domains using side information,” *IEEE Access*, vol. 8, pp. 141 910–141 922, 2020. 28

[91] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, “Iron: Private inference on transformers,” in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=deyqjpcTfsG 30

[92] S. Adams, D. Melanson, and M. De Cock, “Private text classification with convolutional neural networks,” in *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, 2021, pp. 53–58. 30, 39, 41

[93] Q. Feng, D. He, Z. Liu, H. Wang, and K.-K. R. Choo, “Securenlp: A system for multi-party privacy-preserving natural language processing,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3709–3721, 2020. 30

[94] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, “Crypten: Secure multi-party computation meets machine learning,” *Advances in Neural Information Processing Systems*, vol. 34, 2021. 30

[95] S. I. Popoola, G. Gui, B. Adebisi, M. Hammoudeh, and H. Gacanin, “Federated deep learning for collaborative intrusion detection in heterogeneous networks,” in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. IEEE, 2021, pp. 1–6. 31

[96] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, “Federated learning for internet of things: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021. 31

[97] A. H. Celdrán, P. M. S. Sánchez, C. Feng, G. Bovet, G. M. Pérez, and B. Stiller, “Privacy-preserving and syscall-based intrusion detection system for iot spectrum sensors affected by data falsification attacks,” *IEEE Internet of Things Journal*, 2022. 31, 54

[98] O. Aouedi and K. Piamrat, “F-bids: Federated-blending based intrusion detection system,” *Pervasive and Mobile Computing*, p. 101750, 2023. 31

[99] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, “Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment,” *Sensors*, vol. 23, no. 13, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/5941 31, 60

[100] J. Li, X. Tong, J. Liu, and L. Cheng, “An efficient federated learning system for network intrusion detection,” *IEEE Systems Journal*, 2023. 31

[101] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117 734–117 745, 2021. 31, 54

[102] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *Journal of Network and Computer Applications*, vol. 66, pp. 1–16, 2016. 31

[103] Y. Cheng, J. Lu, D. Niyato, B. Lyu, J. Kang, and S. Zhu, "Federated transfer learning with client selection for intrusion detection in mobile edge computing," *IEEE Communications Letters*, vol. 26, no. 3, pp. 552–556, 2022. 32

[104] E. M. Campos, P. F. Saura, A. González-Vidal, J. L. Hernández-Ramos, J. B. Bernabé, G. Baldini, and A. Skarmeta, "Evaluating federated learning for intrusion detection in internet of things: Review and challenges," *Computer Networks*, vol. 203, p. 108661, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621005405 32

[105] Y. Qin and M. Kondo, "Federated learning-based network intrusion detection with a feature selection approach," in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2021, pp. 1–6. 32

[106] L. Mokry, P. Slife, P. Bishop, J. Quiroz, C. Guzzi, Z. Chen, A. Crainiceanu, and D. Needham, "Efficient and privacy-preserving collaborative intrusion detection using additive secret sharing and differential privacy," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 3324–3333. 32, 53

[107] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, 2022. 32

[108] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, and R. Zhang, "A hybrid approach to privacy-preserving federated learning," *CoRR*, vol. abs/1812.03224, 2018. [Online]. Available: http://arxiv.org/abs/1812.03224 32

[109] P. M. S. Sánchez, A. H. Celdrán, T. Schenk, A. L. B. Iten, G. Bovet, G. M. Pérez, and B. Stiller, "Studying the robustness of anti-adversarial federated learning models detecting cyberattacks in iot spectrum sensors," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 2, pp. 573–584, 2022. 32

[110] 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*, 2015, pp. 1322–1333. 32, 52, 56

[111] M. Mansouri, M. Önen, W. B. Jaballah, and M. Conti, "Sok: Secure aggregation based on cryptographic schemes for federated learning," *Proceedings on Privacy Enhancing Technologies*, 2023. 32, 51

[112] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432. 32

[113] 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*, 2017, pp. 1175–1191. 33

[114] H. Li, Q. Ye, H. Hu, J. Li, L. Wang, C. Fang, and J. Shi, "3dfed: Adaptive and extensible framework for covert backdoor attack in federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1893–1907. 33

[115] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1354–1371. 33

[116] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, H. L. Kwing, T. Parcollet, P. P. d. Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020. 34, 44, 46

[117] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose *et al.*, "Pysyft: A library for easy federated learning," *Federated Learning Systems: Towards Next-Generation AI*, pp. 111–139, 2021. 34

[118] N. Holohan, S. Braghin, P. Mac Aonghusa, and K. Levacher, "Diffprivlib: the IBM differential privacy library," *ArXiv e-prints*, vol. 1907.02444 [cs.CR], Jul. 2019. 44, 45, 46

[119] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023. 46

[120] V. Aggarwal, V. Gupta, P. Singh, K. Sharma, and N. Sharma, "Detection of spatial outlier by using improved z-score test," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 2019, pp. 788–790. 47, 51

[121] M. Guarascio, N. Cassavia, F. S. Pisani, and G. Manco, "Boosting cyber-threat intelligence via collaborative intrusion detection," *Future Generation Computer Systems*, vol. 135, pp. 30–43, 2022. 53, 54

[122] E. Hooper, "An intellilgent infrastructure strategy to improvilng the performance and detection capability of intrusion detection systems," in *2006 Securecomm and Workshops*. IEEE, 2006, pp. 1–15. 54

[123] C. F. T. Pontes, M. M. C. de Souza, J. J. C. Gondim, M. Bishop, and M. A. Marotta, “A new method for flow-based network intrusion detection using the inverse potts model,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1125–1136, 2021. 54

[124] K. V. Jönsson, G. Kreitz, and M. Uddin, “Secure multi-party sorting and applications,” Cryptology ePrint Archive, Paper 2011/122, 2011, https://eprint.iacr.org/2011/122. [Online]. Available: https://eprint.iacr.org/2011/122 54

[125] C. Song, T. Ristenpart, and V. Shmatikov, “Machine learning models that remember too much,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 587–601. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3133956.3134077 56, 65

[126] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, “The secret sharer: Evaluating and testing unintended memorization in neural networks,” in *USENIX 2019*, 2019, pp. 267–284. 56, 65

[127] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713. 68

[128] M. Keller and K. Sun, “Secure quantized training for deep learning,” Cryptology ePrint Archive, Paper 2022/933, 2022, https://eprint.iacr.org/2022/933. [Online]. Available: https://eprint.iacr.org/2022/933 68, 72