Received 18 February 2025, accepted 27 March 2025, date of publication 2 April 2025, date of current version 11 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3557228

SURVEY

GuardianML: Anatomy of Privacy-Preserving Machine Learning Techniques and Frameworks

NGES BRIAN NJUNGLE[1], ERIC JAHNS[1], (Graduate Student Member, IEEE), ZHENQI WU[1], LUIGI MASTROMAURO[1], MILAN STOJKOV[2], AND MICHEL A. KINSY[1], (Member, IEEE)
[1]STAM Center, Ira A. Fulton Schools of Engineering, Arizona State University, Tempe, AZ 85281, USA
[2]Faculty of Technical Sciences, University of Novi Sad, 21000 Novi Sad, Serbia

Corresponding author: Nges Brian Njungle (nnjungle@asu.edu)

ABSTRACT Machine learning has become integral to our lives, finding applications in nearly every aspect of our daily routines. However, using personal information in machine learning applications has raised concerns about user data privacy and security. As concerns about data privacy grow, algorithms and techniques for achieving robust privacy-preserving machine learning (PPML) have become a pressing technical challenge. PPML aims to safeguard the confidentiality of both data and models and ensure that sensitive information remains protected during training and inference processes. Various techniques, protocols, libraries, and frameworks have been proposed for PPML, but choosing the right combination along with the appropriate algorithmic or system parameters for a specific deployment instance can be very difficult. In this work, we introduce GuardianML, an open-source recommendation system for selecting the correct parameters and suitable framework for specific use cases of PPML. GuardianML allows users to search through a wide range of PPML frameworks, techniques, protocols, libraries, and more based on a set of objectives. GuardianML filters potential frameworks based on user-defined criteria, such as the number of parties involved in multi-party computation or the need to minimize communication costs in homomorphic encryption scenarios. The system's recommendations and optimizations are formulated as a maximization problem using linear integer programming to identify the most suitable solution for various use cases. Moreover, this work thoroughly analyzes and presents seventy relevant frameworks in the system's database. Additionally, we offer an open-source repository containing practical examples and documentation for some of the frameworks.

INDEX TERMS Differential privacy, federated learning, homomorphic encryption, multi-party computation, privacy preserving machine learning, trusted execution environment.

I. INTRODUCTION

Machine Learning (ML) has intense applications across every industry, from healthcare, finance, education, government and public services, transport and logistics, retail and e-commerce, and even cybersecurity [1]. In healthcare, ML is used pervasively for disease diagnosis and prognosis, drug discovery and development, treatment recommendation, health monitoring, wearable devices, and genetic studies [2]. With this prevalent use of ML, data privacy and preservation have become a major concern as ML models generally require large amounts of data to train. Sometimes, this data even comes from multiple data sources. While the data providers are interested in protecting their data, the model providers are also interested in protecting their models. The area concerned with protecting and preserving data privacy in ML is generally referred to as Privacy Preserving Machine Learning (PPML) [3]. PPML is a set of rules, techniques, and protocols used to preserve and protect the privacy and confidentiality of customers' information and machine learning models while enabling collaboration and productivity scenarios. It allows organizations to leverage the power of ML while keeping sensitive information safe, preserving confidentiality, encouraging data sharing and collaboration, and complying with the law in some cases. Five main techniques are used for PPML, which include Federated Learning, Multi-party Computation,

The associate editor coordinating the review of this manuscript and approving it for publication was Sangsoon Lim.

Homomorphic Encryption, Differential Privacy, and Trusted Execution Environments [4]. Typically, these techniques use different protocols, libraries, and frameworks to achieve their fundamental promise. Thus, they have different strengths and weaknesses, performing differently in different application scenarios.

One key challenge is determining how to evaluate the numerous proposed PPML tools, select the most suitable ones, and rank them for every specific use case as we adopt PPML. In this study, we analyze the different techniques and frameworks for PPML. To do this, we construct fine-grained criteria for analyzing PPML frameworks. These criteria help in capturing the most essential features of every framework. We then propose GuardianML, a recommendation system that uses the information obtained from our analysis to propose the best-fitted frameworks for every use case. Our specific contributions in this work are as follows:

- We conduct an in-depth analysis of various PPML techniques, capturing the most important features across all PPML frameworks.
- We propose a PPML recommendation system tailored for every application use-case by analyzing each scenario considering factors such as computational efficiency, accuracy, usability, and scalability.
- We offer an open-source repository of containerized PPML frameworks with documentation, making it easy for developers and researchers to use these frameworks.

This work continues in Section II, where we discuss related works. In Sections III, IV, V, VI, VII, and VIII, we introduce the different PPML techniques with their most prominent frameworks and provide a comprehensive comparison between the frameworks of every technique. In Section IX, we propose our PPML recommendation system. We then discuss the features and algorithms while providing abstract exemplary use cases of the system. We then move on to explore several practical use cases of GuardianML in Section X. For each scenario, we present the top five frameworks recommended by the engine. We analyze these results, explaining why they are accurate based on the data provided by the various frameworks outlined in the Background sections. We conclude in Sections XI and XII with discussions, open questions, and future research directions.

II. RELATED WORKS

In recent years, numerous surveys and Systematization of Knowledge (SOK) papers have explored PPML as it gains prominence in industry and academia. Tanuwidjaja et al. [5] provided a comprehensive survey on deep learning techniques needed to access sensitive data. ohra et al. [6] also discussed the different PPML techniques, the libraries involved with each technique, and how PPML applications can be developed with these libraries to ensure specific security guarantees. Boulemtafes et al. [7], Al-Rubaie and Chang [8], Xu et al. [3] all discussed PPML techniques, challenges, threats, and possible enhancements. These generalized works on PPML give an overview of the subfield, but none of them discuss frameworks and how they can be adapted for practical purposes.

Pulido-Gaytan et al. [9] reviewed the application of Fully Homomorphic Encryption (FHE) in PPML, while Podschwadt et al. [10] focused on deep learning architectures with FHE. Pulido-Gaytan et al. [9] showed the interaction between FHE and ML from both a theoretical and practical point of view, studying the state-of-the-art approaches used in developing PPML systems. On the other hand, Podschwadt et al. [10] reviewed scientific publications in deep learning with FHE. The work analyzes the ML models and architectures, bringing out all changes made by authors for ML-FHE compatibility and their performance impact. While Pulido-Gaytan et al. [9] showed some PPML frameworks developed with FHE, the work does not cover the practical usability of the frameworks. On the other hand, Podschwadt et al. [10]focuses mainly on how FHE can be used for PPML. Bellini et al. [11] discussed Multi-party Computation (MPC) and ML, analyzing the different approaches used to combine MPC with ML to offer training and inference privacy. Also, Zhang et al. [12] surveyed research works based on MPC providing training and inference in PPML. The authors categorize the publications into two techniques depending on their linear and non-linear computations approach. While Bellini et al. [11] and Zhang et al. [12] provide insights on MPC in PPML, they do not compare the frameworks for practical purposes. Zhang et al. also [13] provided a comprehensive survey on federated learning, systematically discussing the existing research works in the field and classifying them into five key areas: data partitioning, privacy mechanism, machine learning model, communication architecture, and systems heterogeneity. While the paper covers recent works, it only classifies them and offers no meaningful comparisons.

Gong et al. [14] provided a survey on differential privacy and ML, categorizing all work under two broad categories per different differential privacy mechanisms. In addition, the authors discuss challenges associated with each approach in terms of model utility, privacy level, and applications. While this work is comprehensive, it only compares frameworks through the differential privacy mechanism. Mo et al. [15] published a SOK on confidential computing with ML. In this work, the authors tackle PPML using Trusted Execution Environments (TEEs) while discussing their confidentiality guarantees, integrity assurances, and drawbacks. Li et al. [16] also provides a comprehensive survey on secure computation using TEEs. The authors classified all works under three broad categories: secure outsourced computation, secure distributed computation, and secure multi-party computation. The authors provided an analysis of the protocols and developed a comparative analysis between the different protocols and the work done on them. None of the surveys that cover TEEs analyzed their frameworks and how they achieve privacy in ML.

In contrast to all the surveys and SOKs, our work covers frameworks in all PPML. At the same time, we introduce a structure that generates common factors within all PPML frameworks regardless of the techniques used. This structure makes it easy for us to quantitatively compare PPML frameworks. Beyond documentation and comparison of frameworks, to the best of our knowledge, we are the first to provide a PPML recommendation system that helps users analyze, rank, and recommend PPML frameworks based on their use cases. We also provide a simplified overview of every PPML framework, which is easy to read but gives users all the essential information needed to make essential decisions about it. Finally, we provide the first comprehensive open-source repository with tested and documented versions of PPML frameworks.

III. FEDERATED LEARNING

Federated learning (FL) is an ML approach that allows collaborative model training across decentralized edge devices or servers without sharing raw data. This approach is beneficial in scenarios where data privacy is a concern on the edge, as it ensures that sensitive information remains on the edge devices [17]. In FL, the process typically starts with a global model distributed across the network's devices. Each device trains the model locally using its data and computes the model updates. These updates are then sent to an aggregator, which combines them to improve the global model [18]. The updated global model is then returned to the devices, and the process iterates until the desired model performance is achieved. One key advantage of FL is its ability to leverage the diverse data available on edge devices, leading to a more robust and generalizable model [17]. On the downside, FL does not offer the high level of security that cryptographic methods offer and is limited to collaborative scenarios. Figures 1. and 2. demonstrates how federated learning works in a centralized and decentralized setting respectively. In the centralized setting, the edge devices send trained weights (W) to a central server for aggregation while in the decentralized setting, the edge devices exchange the gradients within themselves. Updated models are then sent back to the users to either continue training or use for local inference.

A. FRAMEWORKS

1) TensorFlow FEDERATED (TFF)

It is an open-source extension of TensorFlow by Google, designed for FL and decentralized ML across distributed data sources. It excels in providing robust simulation tools for thorough testing and validation of federated algorithms, ensuring compatibility with TensorFlow workflows and reducing deployment risks. Its focus on research and experimentation limits its optimization for large-scale production deployments, potentially posing challenges when scaling models to different scenarios [19].

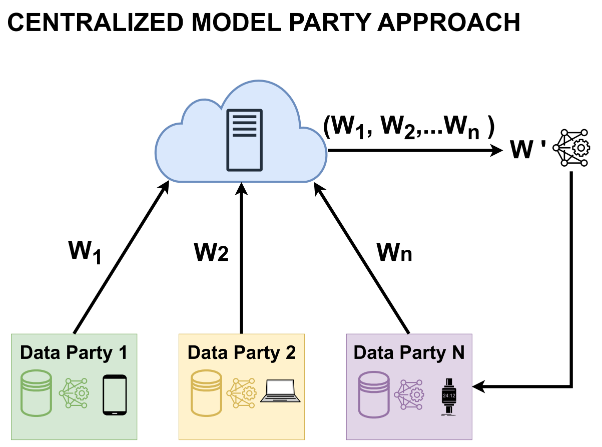


FIGURE 1. Demonstration of privacy-preserving machine learning using centralized federated learning.

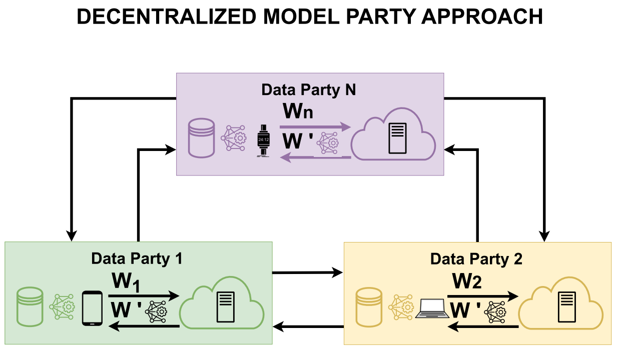


FIGURE 2. Demonstration of a privacy-preserving machine learning using decentralized federated learning.

2) FATE (FEDERATED AI TECHNOLOGY ENABLER)

It is an open-source framework developed by Webank's AI Department, designed to enable FL while ensuring data privacy and security across organizations. It offers a comprehensive toolkit, including FederatedML for algorithms, FATE Flow for scheduling, FATE Board for visualization, FATE Serving for high-performance inference, EggRoll for distributed computation, and KubeFATE for Kubernetes deployment. FATE's main advantage is its strong privacy-preserving capabilities, making it ideal for sensitive industries like finance and healthcare. However, its complexity can be a drawback, posing challenges in deployment and configuration for organizations with limited technical resources [20].

3) FLOWER

It is an open-source Python framework that simplifies the development and deployment of FL systems by providing a high-level API and supporting multiple backends like TensorFlow and PyTorch. It includes essential components for managing federated data, coordinating model updates, aggregating results, and tools for secure communication, monitoring, and visualization. Flower's main advantage is its ease of use and flexibility, making it ideal for developers new

TABLE 1. Summary of federated learning libraries and frameworks. All FL frameworks are open source, support training, inference, and protect both data privacy and model privacy. FL models can support all types of non-linear function.

Framework	ML Operations	Threat Model	Dataset	Inference Time	Accuracy
Tensor Flow federated, [19]	Conv, Linear	Semi-Honest	Small Signal Dataset, Image Set	23s, 58mins	96%, 87%
FATE [20]	Linear, SVM, Clustering	Semi-Honest	UCI datasets, LIBSVM	×	×
Flower [21]	Conv, Linear	Semi-Honest	CIFAR-10, FEMNIST, ImageNet	×	67%, 65%, 80.4%
FLUTE [22]	Conv, Linear, RNN	Semi-Honest, Malicious	MNIST, Fed MNIST, Fed CIFAR-100, Fed Shakespeare	95s, 502s, 6121s, 1310s	81%, 83%, 33%, 57%
FedML [23], [22]	Conv, Linear, RNN	Semi-Honest	MNIST, Federated MNIST, CIFAR-100, CINIC-10	189s, 12754s, 25200s, 9499s	93.19%, 83.0%, 82.57% 81.0%
Leaf [24]	Conv, Linear, LSTM	Semi-Honest	FE-MNIST, Sent140, Shakespeare, CelebA	×	×
IBM Federated Learning [25]	Conv, Linear, Decision Trees	Semi-Honest	MNIST, Adult, Cifar-10, FE-MNIST	×	×
OpenFL [26]	Conv, Linear, Clustering, RNN	Semi-Honest	MNIST	×	×
NVIDIA Clara Federated Learning [27], [28]	Conv, Linear	Semi-Honest	420 portal-venous phase abdominal CT images	×	82.3%
Substra [29]	Conv, Linear	Semi-Honest	MNIST, Cyclic MNIST, IRIS dataset, Diabetes dataset	×	95%, 96%, 100%
FedScale [30]	Conv, Linear	Semi-Honest	FEMNIST, OpenImage, Amazon Review, Reddit	×	78.5%, 75.3%, 76.6%, 63.37%
FederatedScope [31]	Conv, Linear, GNN, BERT	Semi-Honest	FE-MNIST, CIFAR-10, Twitter dataset	×	85%, 70%, 69%

TABLE 2. Specific federated learning features in frameworks. Federated Learning(Fed). Average(Avg). Stochastic Gradient Decent (SDG). Evaluation(Eval). Verified (Ve). Aggregation(Aggr).

Framework	Clients	Rounds	Acceleration	Centralized	Decentralized	Algorithms/Strategies
TensorFlow federated [19]	×	20	×	✓	×	FedAvg, FedSGD, FedEval
FATE [20]	×	×	×	✓	✓	FedAvg, VeFed
Flower [21]	1000	60	GPU	✓	✓	FedAvg, FedProx, FedOptim
FLUTE [22]	3400	4000	GPU	✓	×	FedAvg, DGA FedAdam
FedML [23]	342477	100	CPU, GPU	✓	✓	FedAvg, FedNAS, VeFed, SplitNN
Leaf [24]	35	40	×	×	✓	FedSGD, FedAvg
IBM Federated [25]	×	×	×	✓	✓	FedAvg, SPAHM, PFNM, Krum
OpenFL [32]	×	×	×	✓	✓	FedTr, FedEval, FedAggr.
NVIDIA Clara [27], [28]	10	×	GPU	✓	✓	FedAgr
Substra [29]	3	3, 6	GPU	×	✓	FedAvg, Scaffold, FedPCA
FedScale [30]	10000	1500	CPU, GPU	×	✓	FedAvg, FedProx, FedYoGi, IID
Federated Scope [31]	660820	×	GPU	✓	×	FedAvg, FedBN, FedEM, Ditto

to FL or those in need of quick FL prototypes. However, its lack of advanced features and optimizations might limit its effectiveness in large-scale production environments [21].

4) FLUTE (FEDERATED LEARNING UTILITIES AND TOOLS FOR EXPERIMENTATION)

It is an open-source framework designed for high-performance research and offline simulations in FL, supporting a wide range of research activities such as optimization and privacy enhancements. It provides tools for rapid prototyping, large-scale simulations, and comprehensive testing of FL scenarios, including handling diverse data distributions and complex model architectures. The primary advantage of FLUTE is its capability for detailed experimentation and high-performance simulations, making it ideal for advanced research. However, its focus on offline simulations limits its applicability in real-time or production environments, requiring additional customization for deployment [22].

5) FedML

It is an open-source research library dedicated to FL, providing various tools and algorithms for developing and experimenting with federated learning systems. It addresses challenges like client heterogeneity and communication efficiency through diverse optimization algorithms and incorporates advanced privacy-preserving techniques to ensure data security during model training. FedML also includes robust evaluation metrics for assessing model performance, and its modular design facilitates integration with existing ML systems. The primary advantage of FedML is its strong research orientation, making it a powerful platform for exploring and advancing FL methodologies. However, it is limited in immediate applicability in production scenarios, necessitating additional customization for large-scale deployment [23].

6) LEAF

Leaf is a lightweight and flexible benchmarking library designed to simplify large-scale FL experiments, making it accessible to researchers and practitioners. It provides essential tools for managing federated data, aggregating model updates from multiple clients, and evaluating key performance metrics such as accuracy, communication efficiency, and convergence rates. Leaf's modular architecture supports the integration of new techniques and algorithms, catering to both small-scale and large-scale FL scenarios. The primary advantage of Leaf is its ease of use and rapid setup, which is ideal for quickly deploying FL experiments. However, its focus on simplicity and benchmarking results in a limited feature set, potentially requiring additional tools or extensions for more advanced applications or production deployment [24].

7) IBM FEDERATED LEARNING

It is an advanced framework integrated with IBM Watson ML, enabling collaborative model training across distributed datasets while preserving data privacy. It allows organizations to train ML models without exchanging raw data, ensuring data confidentiality. The framework offers tools for model training, deployment, and monitoring and integrates seamlessly with IBM's AI ecosystem, providing robust support for FL workflows. Its primary advantage is its ability to maintain data security while benefiting from IBM's enterprise-level support, making it ideal for large-scale applications. However, its complexity and resource demands make it challenging for smaller organizations or projects, and its reliance on IBM's ecosystem also limit flexibility for users needing a broader range of tools, [25].

8) OpenFL (OPEN FEDERATED LEARNING LIBRARY)

OpenFL is an open-source library by Google that extends TensorFlow for FL and decentralized ML applications. It provides abstractions for federated computations, supporting the development and testing of algorithms on distributed data sources. It includes simulation tools for evaluating and optimizing FL algorithms and supports various optimization, privacy, scalability, and communication scenarios. Its key advantage is robust integration with TensorFlow, ensuring compatibility and easing transitions to FL. However, its dependence on TensorFlow may limit flexibility for users preferring other frameworks or those unfamiliar with TensorFlow, potentially reducing its adaptability for diverse applications [26]

9) NVIDIA CLARA FEDERATED LEARNING

It is a platform designed for secure AI model training across distributed data sources, focusing on healthcare applications. It enables collaborative development of diagnostic tools by allowing hospitals to train local models on private data and share only aggregated updates, ensuring data privacy. Clara supports various FL structures and uses gRPC for encrypted communication, with tools for managing training processes and model performance. Its main advantage is its specialized approach to healthcare, maintaining stringent data privacy. However, its healthcare focus limits applicability in other sectors, and reliance on NVIDIA's ecosystem also restricts usage in organizations using different ML infrastructures [33].

10) SUBSTRA

It is a framework designed to enhance privacy, traceability, and collaboration in ML through a decentralized client network. It facilitates secure collaboration by exchanging algorithms, models, and metadata without transferring sensitive data, using Distributed Ledger Technology (DLT) to maintain a transparent, immutable record of activities. Originally developed for healthcare, Substra's robust privacy and traceability features support various domains and FL scenarios. Its main advantage is the strong privacy protection and collaboration it enables through DLT. However, its complexity in implementation and management, due to DLT and decentralized systems, may be a disadvantage due to the scalability, performance, and resource requirements, complicating deployment and maintenance [29].

11) FedScale

It is a FL framework designed for benchmarking and reproducible research. It offers structured APIs, runtime management tools, and access to 20 realistic datasets from various domains like video streaming and image processing. FedScale ensures client privacy and supports simulations of heterogeneous environments, facilitating both on-device and large-scale experiments for comprehensive performance evaluation and scalability testing. Its main advantage is the extensive suite of datasets and structured APIs that enable thorough benchmarking and research. However, its focus on experimentation limits its applicability for production use, and the complexity of managing multiple datasets and simulations might require significant resources and expertise [30].

12) FederatedScope

This FL framework has an event-driven architecture designed to address the complexities of diverse participant characteristics, including data, resources, and learning objectives. It offers extensive features for managing FL courses and supports synchronous and asynchronous training modes, allowing for flexible and sophisticated functionalities integration. FederatedScope main limitation is its adaptability and robust programming interfaces, which enable effective handling of varied use cases in both academic and industrial settings. However, its complexity and steep learning curve pose challenges for real-world setup and management, potentially requiring significant resources and expertise to fully utilize the framework [31].

Table 1 shows a comparison between all FL frameworks using generalized features. We identified ten common factors across all PPML frameworks that give a complete overview of every framework. These factors include Opensource state, ML Operations, Threat Model, Non-linear functions, Training Support, Datasets, Data privacy, Model Privacy, Inference Time, and Model Accuracy. The ML operations, datasets, and non-linear functions are essential for creating relations between use cases and frameworks. The threat model, training and inference support, data and model privacy, open-source state, and inference time contribute beyond ties as they are also used for quantitative comparisons of the frameworks. Opensource state, training, inference support, data, and model privacy are boolean since a framework can either support or not support them. In all Tables, ✓ and × are used for supported and not-supported, respectively. Data Privacy means that data is kept private during training or inference. Model privacy means the ML model is kept private for training or inference. The threat models, accuracy, and inference time, are extracted from the publication of the framework. For frameworks that present more than four pieces of information for a single feature in its publication, we select and show the top four results in this paper. However, we insert all information on the frameworks in the database used for the recommendation system proposed in this work. ML operations show the different learning layers and ML models supported. Non-linear stands for Non-linear functions and it shows all non-linear functions implemented in case the technique does not support standard non-linear functions. In the tables, we will represent data as follows: Convolution (Conv), Support Vector Machine (SVM), Recurrent Neural Network (RNN), Graph Neural Network (GNN), and Traditional Non-linear functions (TradNL).

Table 2 summarizes FL frameworks using federated learning-specific features across the frameworks focusing on the number of clients and rounds they support and have been tested for the framework respectively whether they leverage hardware acceleration such as GPU supported while centralized, decentralized feature shows whether the framework supports the specified type of FL or both. Finally, algorithms highlight the algorithms or strategies each framework supports. All abbreviations not defined are standardized names in ML.

IV. MULTI-PARTY COMPUTATION

Secure Multi-Party Computation (MPC) is a cryptographic protocol that enables a group to jointly perform a computation without disclosing any participant's private inputs [34]. Andrew Yao laid the foundations of MPC with his work on building computationally secure protocols in 1982 [35]. He introduced what is known as the Garbled Circuit Protocol, which allows a set of parties to jointly compute a function without exposing their private inputs. The canonical example given as motivation in Yao's work is known as Yao's millionaire's problem [35]. In this problem, two millionaires seek to determine who is richer without leaking their net worth to one another. Since the inception of Garbled Circuits, many other MPC protocols have emerged with various security guarantees, computational complexity, and communication overhead. A general illustration of MPC applied to private inference or training between two parties can be seen in Figure 3. This figure shows a data and model party looking to perform private inference or training. The first step of this is for each party to generate secret shares of their data and send it to the corresponding party. After the secret shares are received, the two parties communicate to privately perform computations on shares. At the end of the communication rounds, the inference or training results will be accessible to a singular party.

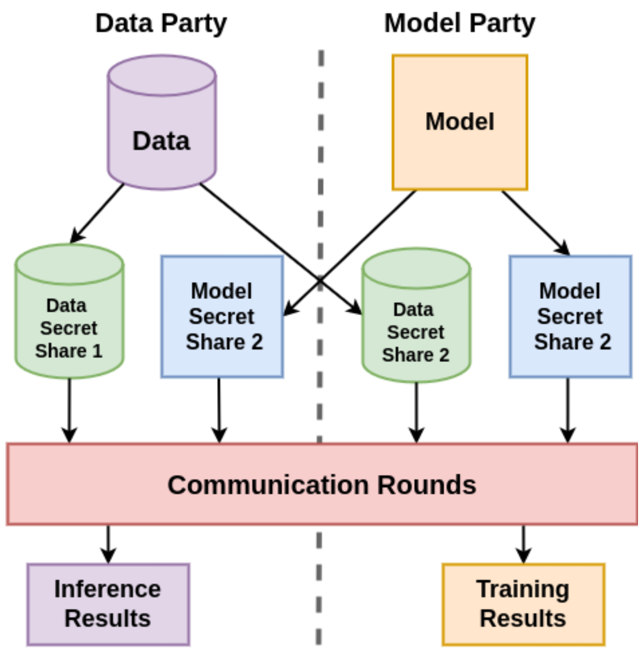


FIGURE 3. Demonstration of a privacy-preserving inference between two parties ensuring both data and model privacy.

A. FRAMEWORKS

1) SecureML (A SYSTEM FOR SCALABLE PRIVACY-PRESERVING MACHINE LEARNING) [36]

The authors introduce SecureML, a 2-party framework enabling two non-colluding parties to train a neural network without revealing private information. The framework consists of a setup phase, where parties encrypt or share data, and a computation phase, where encrypted data is used for training or inference. Several optimizations reduce latency during the online phase, including the use of multiplication triples for accurate fixed-point operations. They also switch between arithmetic secret sharing and Yao's Garbled Circuits, utilizing Oblivious Transfer for efficient logistic and softmax approximations. Their vectorized implementation significantly speeds up training, achieving 93.4% accuracy on MNIST with an MLP, only 1% lower than a non-privacy-preserving model. This work does provide the first implementation of PPML training. However, they heavily rely on an expensive, offline, preprocessing phase, sometimes requiring gigabytes of communication.

2) MiniONN (OBLIVIOUS NEURAL NETWORK PREDICTIONS VIA MiniONN TRANSFORMATIONS) [37]

It is a 2-party framework that transforms existing neural networks into oblivious equivalents for PPML inference, requiring no modifications to training or pre-trained networks. In MiniONN, a server and client hold secret input and output values shares. A preprocessing phase generates multiplication triples using homomorphic encryption to speed up operations like dot products. Secret sharing allows for efficient linear transformations. MiniONN achieves 97.6% accuracy on MNIST with a LeNet-5-like architecture and 81.61% on CIFAR-10 with a 10-layer CNN. It also demonstrates a 230-fold latency reduction and 8-fold message size reduction compared to SecureML. It drastically decreases latency and message size compared to previous works but still has issues scaling to larger models.

3) CHAMELEON (A HYBRID SECURE COMPUTATION FRAMEWORK FOR MACHINE LEARNING APPLICATIONS) [38]

Riazi et al. [38] propose Chameleon, a hybrid inference framework that protects against semi-honest parties. Chameleon combines Garbled Circuits (GC) and additive Secret Sharing (SS) in an online phase and uses a Semi-honest Third Party (STP) in an offline phase. During the offline phase, Oblivious Transfers (OT), arithmetic and boolean multiplication triples, and vector shares are precomputed, with the STP generating correlated randomness. In the online phase, linear operations utilize additive SS, while non-linear operations use either GC or MPC based on efficiency. Evaluation of MNIST shows no accuracy loss and improved communication and computational efficiency compared to state-of-the-art methods. This work further improves communication and computation efficiency but fails to evaluate their work on larger models.

4) ABY3 (A MIXED PROTOCOL FRAMEWORK FOR MACHINE LEARNING) [39]

ABY3 is a three-party protocol that protects against malicious adversaries. This system uses three servers, each given secret data shares before jointly training and inferencing. They use approximate fixed-point multiplication and piecewise polynomial functions to obtain better approximations of non-linear functions. They compute these functions using Boolean, Arithmetic, and Yao's circuits. When training a network on MNIST with their framework, they use a CNN architecture, which achieves 99% accuracy on MNIST while being up to 55k× faster than state-of-the-art (SOTA). This work can achieve SOTA accuracy on simple datasets, such as MNIST, but fails to show any scalability.

5) EzPC (PROGRAMMABLE AND EFFICIENT SECURE TWO-PARTY COMPUTATION FOR MACHINE LEARNING) [41]

This work presents EzPC, a secure 2PC framework. EzPC provides formal correctness and security guarantees while maintaining performance and scalability. They have a compiler to generate protocols that combine arithmetic and boolean circuits. They demonstrate up to 19× performance over SOTA. The custom compiler allows for various optimizations to be made for both computation and communication but does not support model training.

6) FLASH (FAST AND ROBUST FRAMEWORK FOR PRIVACY-PRESERVING MACHINE LEARNING) [40]

It is a 4PC framework with protection against one malicious party. It provides guaranteed output delivery irrespective of adversary behavior using fixed-point techniques to evaluate circuits along with special MSG extraction and MAX comparison for non-linear evaluation. They observe 155× and 8.5× speed-ups over ABY3 in LAN and WAN, respectively. This work employs a 4-party setting which allows for increased computational efficiency but results are only showcased on small model architectures.

7) CRYPTFLOW (SECURE TensorFlow INFERENCE) [42]

This work presents CrypTFlow, which converts TensorFlow inference code into MPC protocols. They include Athos, an end-to-end compiler from TensorFlow to various semi-honest MPC protocols. Porthos, a semi-honest 3PC for speedups in TensorFlow applications. Finally, Aramis provides hardware and integrity guarantees to convert semi-honest protocols to malicious security. They are inference on ResNet-50 and Densenet-121 in under 30 seconds in a semi-honest setting and under 2 minutes in a malicious setting. This work reduces the overhead of converting models to privacy-preserving equivalents and showcases large model inferences. This framework only supports conversion from TensorFlow models, restricting users from other popular frameworks.

8) BLAZE (BLAZING FAST PRIVACY-PRESERVING MACHINE LEARNING) [43]

It is a PPML framework for three parties, tolerating one malicious adversary while improving ABY3. They build primitives such as multiplication, bit extraction, and bit-to-arithmetic sharing conversion as they rely heavily on secret sharing over arithmetic and boolean fields. By adding an input-independent preprocessing phase, they can outperform ABY3 and FLASH by 4-3600×. It uses a three-server setting tolerating one malicious corruption over a ring. It has an input-independent preprocessing phase, a fast input-dependent online phase relying on efficient PPML primitives such as (i) A dot product protocol and (ii) A truncation method. While this work develops one of the quickest preprocessing phases compared to previous works, they only validate their results using linear/logistic regression and a

TABLE 3. Summary of multi-party computation frameworks. All MPC frameworks support inference, data privacy and model privacy. Abbreviations: Semi-Honest Third Party (STP).

Framework	Opensource	ML Operations	Threat Model	Non-Linear	Training	Dataset	Inference Time	Accuracy
MiniONN [37]	×	Conv, Linear, LSTM	Semi-Honest	Piece-wise Polynomial	×	MNIST, CIFAR-10, Pen TreeBank	9.5s, 544s, 18s	99.0%, 81.61%, ×
SecureML [36]	✓	Linear	Semi-Honest	Low-Degree Polynomial	✓	MNIST, Gisette, Arcene	294,239.7s, 268.9s,	98.64%, 97.9%, 86.0%
Chameleon [38]	×	Conv, Linear	Semi-Honest + STP	TradNL	×	MNIST, CIFAR-10	2.24s, 52.67s	99.0%, 81.61%
ABY3 [39]	✓	Conv, Linear	Semi-Honest, Malicious	Piece-wise Polynomial	✓	MNIST	1 hour	99.0%
FLASH [40]	×	Linear	Semi-Honest, Malicious	TradNL	✓	MNIST	×	×
EzPC [41]	✓	Conv, Linear	Semi-Honest	TradNL	×	MNIST, CIFAR-10	90.8s, 647.5s	99.2%
CryptFlow [42]	✓	Conv, Linear	Semi-Honest, Malicious	TradNL	×	ImageNet	36.0s	93.23%
BLAZE [43]	×	Linear	Semi-Honest, Malicious	TradNL	✓	Parkinson	×	×
Tetrad [44]	×	Conv, Linear, SVM	Malicious	Piece-wise Polynomial	✓	MNIST, CIFAR-10	0.26s, 85.69s,	×
SWIFT [45]	×	Conv, Linear	Honest-Majority	TradNL	✓	MNIST, CIFAR-10	5.08s, 15.89s	×
Aegis [46]	×	Conv, Linear	Semi-Honest	TradNL	×	×	×	×
Secret Flow-SPU [47]	✓	Conv, Linear, LSTM	Semi-Honest, Malicious	Newton-Raphson Approximation	✓	MNIST, LSTM Sine Regression	9131s, 3.01s	×

TABLE 4. MPC framework with schemes and number of parties supported. Oblivious Transfer (OT). Homomorphic Encryption (HE). Garbled Circuits (GC). Secret Sharing (SS). Arithmetic Circuits (AC). Boolean Circuits (BC).

Framework	Schemes	Participants
MiniONN [37]	HE, SS	2-Party
Chameleon [38]	SS, GC, GMW	3-Party
SecureML [36]	SS, BC, GC	2-Party
ABY3 [39]	SS, AC, BC, GC	3-Party
FLASH [40]	SS, AC, BC	4-Party
EzPC [41]	SS, AC-BC Combination	2-Party
CryptFlow [42]	SS, AC	3-Party
BLAZE [43]	SS, AC, BC	3-Party
Tetrad [44]	SS, AC, BC, GC	4-Party
SWIFT [45]	SS, AC, BC	(3,4)-Party
Aegis [46]	SS, AC	3-Party
Secret Flow-SPU [47]	SS, AC, BC, GC	N-Party

multi-layer perception which fails to show the scalability of their method.

9) TETRAD (ACTIVELY SECURE 4PC FOR SECURE TRAINING AND INFERENCE) [44]

The authors propose Tetrad as an enhanced framework that supports four parties and protects against one corrupted participant. Tetrad offers various security configurations and improves arithmetic and boolean circuits, reducing the required elements while maintaining security. It enables probabilistic truncation without added costs and minimizes communication rounds through multi-input multiplication gates. Additionally, Tetrad utilizes mixed protocols across different circuit types, significantly increasing throughput while providing better security, tested on LeNet-5 and VGG-16 with MNIST and CIFAR-10, respectively. This work supports conversion between boolean, arithmetic, and garbled circuits with heavily decreased communication requirements compared to ABY3 and others. Tetrad can be used with larger model architectures but does not support training algorithms such as ResNets and LSTMs as Tetrad doesn't support batch normalization.

10) SWIFT (SUPER-FAST AND ROBUST PRIVACY-PRESERVING MACHINE LEARNING) [45]

SWIFT is a robust PPML framework for a range of ML algorithms in a Secure Outsourced Computation paradigm. SWIFT is a three- or four-party maliciously secure framework. It guarantees output delivery irrespective of adversarial behavior. SWIFT is the first robust and efficient PPML framework in a 3 setting and outperforms other work, such as BLAZE and FLASH, at a higher security level. This work does not support mixed-world conversions.

11) AEGIS (A LIGHTNING FAST PRIVACY-PRESERVING MACHINE LEARNING PLATFORM AGAINST MALICIOUS ADVERSARIES) [46]

This work proposes a PPML built on top of a maliciously secure 3PC framework over a ring. They reduce communication by 75% compared to SOTA in semi-honest settings and by 50% in malicious settings. They also show their ReLU and MaxPool PPML protocols outperform SOTA by 4x and 10x for semi-honest and malicious settings. This work spends time making algorithmic improvements to multiplication, ReLU, MaxPooling, etc. which all scale better with input size. However, this work fails to provide accuracies for their trained models.

12) SecretFlow-SPU (A PERFORMANT AND USER-FRIENDLY FRAMEWORK FOR PRIVACY-PRESERVING MACHINE LEARNING) [47]

It is a high-performing PPML framework developed to help people with little knowledge of cryptography develop applications in a privacy-preserving manner. It consists of a compiler frontend that takes an ML program and converts it into an MPC-specific intermediate representation. The code is then optimized and run through different MPC protocols. SecretFlow-SPU supports simple conversion of existing neural networks built in JAX using its custom compiler. Its implementation only supports fixed-point numbers, which can affect operations such as random number generation.

Table 3 compare MPC frameworks using generalized features while Table 4 compares MPC frameworks and their schemes and number of parties involved.

V. HOMOMORPHIC ENCRYPTION

Homomorphic Encryption(HE) can be defined as an advanced cryptographic protocol that enables computations on encrypted data. The principle was first introduced in 1978 as *Privacy Homomorphism* [48]. Still, it was not until 2009 that Craig Gentry demonstrated the first Fully Homomorphic Encryption scheme allowing for arbitrary homomorphic computations on ciphertext by applying a principle called Bootstrapping [49]. Formally, if given two messages m_1 and m_2 with an encryption function **Encrypt**, and computationally feasible functions f and f', f and f' are said to be homomorphic if:

$$f(m_1, m_2) = f'(\mathbf{Encrypt}(m_1), \mathbf{Encrypt}(m_2)) \qquad (1)$$

CKKS [50], BFV [51] BGV [52] and TFHE [53] are the main HE schemes used in PPML today. Due to the complex nature of schemes, FHE Libraries such as SEAL [54], OpenFHE [55], HEAAN [50], TFHE-rs [56], and HElib [57] are been used to leverage to take an advantage of the implementation of this schemes. Figure 4. shows a scenario where HE is used to secure both model and data privacy in machine learning inference using an untrusted third party in form of cloud server. The encrypted FHE Model is placed in the server, and clients can send encrypted messages for inference into the server. The Server does not decrypt this message but instead inference them in their encryted format. The encrypted results are then sent back to the client who then decrypts it and get the right results.

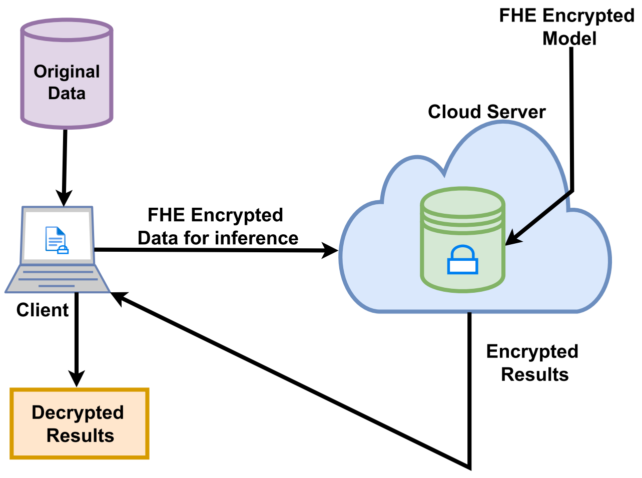


FIGURE 4. Demonstration of a privacy-preserving inference on an encrypted ML model with encrypted data via Homomorphic Encryption using an untrusted third party.

A. FRAMEWORKS

1) CryptoNets [58] AND FASTER CryptoNets [71]

Dowlin et al. [58] proposed CrytoNets as the first full-blown framework using FHE in Neural Networks Inferencing. CryptoNets is built on SEAL and implemented in C#. It initially implemented an FHE scheme called YASHE based on NTRU with a later version called Faster Cryptonets based on BFV scheme in SEAL [71]. It uses a polynomial activation function, and it shows that using a batch normalization layer before each activation layer stabilizes training with FHE. The solution relied heavily on single instruction multiply Data (SIMD) batching to improve computational overhead, thus enabling the efficient evaluation of large batches of data. The main disadvantages of CryptoNets are that they do not support training, have huge computation overheads, and are limited for use on Windows systems.

2) E2DM [59]

It stands for encrypted data and encrypted model [59]. It is a privacy-preserving neural network for inferencing based on Jiang et al.'s matrix multiplication algorithm built on CKKS. It uses HEAAN, the original CKKS library, and has a computational complexity of $O(d^2)$. It uses the SIMD batching technique for high performance. The framework also implements an advanced bootstrapping technique in SEAL to evaluate deep neural networks. This is the first work that actively supports model training with an improved version of matrix multiplication with FHE. However, it is not actively maintained and has limited documentation.

3) PrivFT [60]

PrivFT provides two solutions for privacy preservation text inferencing: using an encrypted dataset to train an encrypted model and using encrypted data to infer on a trained model [60]. It relies on the implementation of CKKS in SEAL. It also implements a special residue number system

TABLE 5. FHE frameworks comparison based on general features. All FHE frameworks support inference and model privacy and protect against semi-honest adverseries.

Framework	Opensource	ML Operations	Threat Model	Non-Linear	Training	Dataset	Data Privacy	Inference Time	Accuracy
Cryptonets [58]	✓	Linear, Conv	Semi-Honest	Square, Polynomials	×	MNIST, CIFAR-10	×	45.7s, 0.9h	99.17%, 75.99%
E2DM [59]	×	Linear, Conv	Semi-Honest	Square	×	MNIST	✓	28.59s	98.1%
PrivFT [60]	×	text classification	Semi-Honest	×	✓	IMDB, AGNews, Yelp, DBPedia	✓	7.9s , 7.8s, 0.65s, 0.63s	91.5%, 92.5%, 96.1% , 98.80%
nGraph-HE [61]	✓	Compiler	Semi-Honest	×	✓	MNIST, CIFAR-10	✓	14.8s, 53.3s	96.9% 68.5%
CHET [62]	×	Compiler	Semi-Honest	×	×	MNIST, CIFAR-10	✓	35.2s, 164.7s	99.3%, 81.5%
CryptoDL [63]	✓	Linear, Conv	Semi-Honest	Approximation	×	MNIST, CIFAR-10	✓	320s, 3.2h	99.56%, 94.2%
PyHENet [64]	×	Linear, Conv	Semi-Honest	Polynomial	✓	MNIST, CIFAR-10	✓	×	99.9%
Lee et al. [65]	×	Linear, Conv	Semi-Honest	Approximation	×	CIFAR-10	✓	3h	98.43%
AHEC [66]	×	Compiler	Semi-Honest	×	✓	×	✓	4.32s	×
Lorenzo et al. [67]	✓	Linear, Conv	Semi-Honest	Approximation	×	CIFAR-10	✓	342s	91.67%
PPDL [68]	×	Linear, Conv	Semi-Honest	Approximation	✓	MNIST	✓	8.8s	97.84%
HCNN [69]	×	Linear, Conv	Semi-Honest	Polynomial	×	MNIST, CIFAR-10	✓	5.1s, 304.43s	99%, 77.55%
TT-TFHE [70]	×	Cov	Semi-Honest	SeLU	×	MNIST, CIFAR-10	✓	0.04s, 0.4s	97.5%, 70.4%

TABLE 6. FHE frameworks comparison based on domain-specific features.

Framework	Library	Scheme	Normalization	Acceleration	Bootstrapping
Cryptonets [58]	SEAL	YASHE, BFV	✓	×	✓
E2DM [59]	HEAAN, SEAL	CKKS	×	×	✓
PrivFT [60]	SEAL	CKKS	×	GPU	✓
nGraph-HE [61]	SEAL	CKKS	✓	×	✓
CHET [62]	HEAAN	CKKS	✓	×	✓
CryptoDL [63]	HELib	BGV	✓	×	✓
PyHENet [64]	×	CKKS	×	GPU	×
Lee et al. [65]	×	CKKS	✓	×	✓
AHEC [66]	SEAL	CKKS	×	GPU	×
Lorenzo et al. [67]	OpenFHE	CKKS	✓	×	✓
PPDL [68]	HElib	CKKS	×	×	×
HCNN [69]	SEAL	BFV	×	GPU	×
TT-TFHE [70]	TFHE-rs	TFHE	✓	×	✓

customized to work with CKKS over GPUs, thus providing a substantial computational speedup. The training process on encrypted data is very expensive, taking up to 5.04 days with GPU to train and achieving a runtime of 0.17 seconds on the Natural Language Processing public dataset.

4) nGRAPH-HE [61]

nGraph-HE is a backend developed for Intel's nGraph compiler that converts neural networks developed with TensorFlow or PyTorch to FHE-friendly versions [61]. It helps users implement privacy-preserving neural networks with very little knowledge of FHE. It uses the CKKS implementation of the SEAL library. The main advantage of this compiler is that it gives users with very little knowledge of FHE the ability to exploit the technology in building PPML applications with extremely user-friendly frontends and minimal overhead. On the downside, it has very limited documentation and relies on an old version of SEAL.

5) CHET [62]

CHET is an FHE domain-specific compiler that optimizes FHE neural network inferencing using the CKKS implementation from SEAL and the original HEAAN implementation [62]. It supports a domain-specific language for specifying tensor circuits, automating many laborious and error-prone tasks that occur during FHE inferencing, such as parameter selection and efficient tensor layout. CHET has an advantage over nGraph-HE in that it automatically determines security parameters and provides a hardware ISA interface that allows hardware advances to proceed independently of software advances. On the downside, it does not seamlessly integrate with already existing frontends, such as TensorFlow, and the benchmark results provided are much slower than those of nGraph-HE for neural networks.

6) AHEC [66]

AHEC is an end-to-end HE compiler for efficient PPML inference. It uses a domain-specific language that enables automatic generation and optimization of HE kernels across a range of hardware platforms and ML frameworks. When used with Cryptonets, it demonstrates a 750-time improvement in CNN inference speed. On the downside, it highly reduces the network's accuracy compared to other compilers like CHET and nGraph-HE.

7) CryptoDL [63]

It relies on the BGV scheme implementation of HElib [63]. This framework is focused on image classification through the implementation and use of well-known CNNs. It starts by designing an approximation of the activation functions commonly used in CNNs as low-degree polynomials, then trains the CNN with these functions on encrypted data. It shows a very high accuracy of 99.52% on MNIST, which is very close to the accuracy of 99.77% on best-performing CNN [63]. It provides high efficiency, accuracy, and stability for PPML predictions. The inference time of CryotoDL is very high compared to SOTA.

8) PyHENet [64]

The PyHENet proposes a different approach to carry out FHE inferencing [64]. In this approach, encrypted data based on the CKKS scheme is uploaded to the cloud. Second, the convolutional computation over floating-point ciphertext data is carried out using SIMD, which ensures low accuracy loss inference with aided parallel computation. Thirdly, inferencing results on ciphertext are sent back to the client for decryption. Here, the authors carry out the second phase in plaintext format using a GPU accelerator while keeping other phases unchanged. The relative accuracy of the network is 100% compared to CNNs trained on plaintext data. The main disadvantage of this framework is that it heavily depends on a high-end third party, such as the cloud.

9) LEE et al. 2021 (PRIVACY-PRESERVING MACHINE LEARNING WITH FULLY HE FOR DEEP NEURAL NETWORK) [65]

This work implements a standard ResNet-20 model using a special residue number system for CKKS with bootstrapping and verifies the implementation on a CIFAR-10 dataset and plaintext model parameters [65]. The implementations use approximation functions to replace non-linear functions such as RELU with sufficient precision. The implemented bootstrapping allows the evaluation of an arbitrary depth neural network on encrypted data. The accuracy of the model is 98.43%, which is equivalent to that of the original model with non-encrypted data. It require high memory and demonstrate a very high inference time compared to other frameworks in the same category.

10) LORENZO et al. 2024 [67]

Lorenzo et al. proposed an encrypted image classification framework with a low memory footprint using FHE. It uses a Residue network implementation of CKKS to develop a circuit that reduces the memory requirements of FHE by more than 85% compared to the most recent works while maintaining a high level of accuracy and a short computational time. It is built on OpenFHE using Chebyshev Polynomials [72] to approximate Non-linear RELU functions. The framework is implemented with CKKS for fast linear layers. It is the most recent open-source FHE framework analyzed, and it demonstrates low memory usage with a high inference accuracy and time compared to other networks using similar approaches, such as [65]. On the downside, it implements just inferencing, and it is not very easy to understand and customize to other architectures.

11) PPDL [68]

The work focuses on exploring classes of FHE in enabling real-world inference of neural networks [68]. It experiments on MNIST and Melanoma datasets using CKKS implementation of the HELib library to achieve an end-to-end encryption inference in a binary classification problem. The inference time of this work is very impressive, but it is not easily reproducible compared to other open-source works.

12) HCNN [69]

This is the first work that uses GPU to carry out HE inference on deep learning. This acceleration provided to FHE-CNN inference leads to high performance on both MNIST and CIFAR-10 with high accuracies. The BFV scheme implementation in SEAL library is used to achieve PPML inference while NTL is used to implement the network used in training. It uses quantized networks with low-precisions for optimal training since the BFV scheme supports only integer evaluations.

13) TT-TFHE [70]

This framework uses a recent type of covolution neural networks called Truth-Table Neural Networks to implement and easy to use CPU- based PPML inference framework based on the TFHE scheme. The solution presents a low memory footprint for both MNIST and CIFAR-10 with high accuracies. It is implemented using Concrete a python framework build on the TFHE-rs framework for evaluation of FHE circuits.

Table 6 compares FHE frameworks based on generalized features while 5 shows a comparison between all ML Frameworks developed based on FHE-specific features. Normalization, Acceleration, and Bootstrapping shows wether the framework supports this operations while Scheme and Library shows the specific HE schemes and library used.

VI. DIFFERENTIAL PRIVACY

Differential Privacy (DP) is used in PPML to prevent malicious parties from exfiltrating private information about the data used to train a model [73]. DP aims to safeguard private information from malicious parties by adding random noise to training data. The protection of this noise is based on the amount added to the system. This usually comes with a trade-off between model performance and privacy. This added noise makes it difficult for adversaries to extract private information from the training data and allows for plausible deniability on the side of the data owner [10]. Unlike many other cryptographic techniques, DP operates on standard data types and is thus easy to implement in modern ML frameworks with support for hardware acceleration. The most considerable downside of DP is that it reduces a model's predictive powers while failing to offer many of the cryptographic guarantees present in other protocols [73]. Differential Privacy does come with several valuable properties, including group privacy, composability, and robustness to auxiliary information [73]. An illustration of Differential Privacy can be seen in Figure 5. The figure shows local differential privacy setting, where parties add deferentially private noise to their local data before sending it to an untrusted data curator. The user party can then directly query the database without additional steps from the database provider.

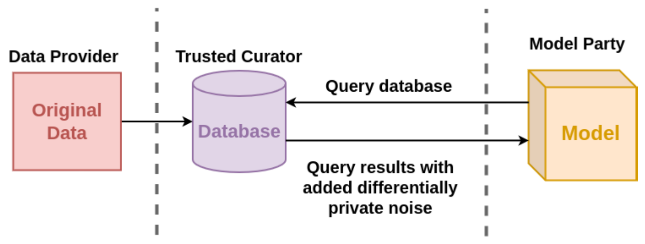


FIGURE 5. Demonstration of a privacy-preserving machine learning using differential privacy.

A. FRAMEWORKS

1) ABADI et al. 2016: DEEP LEARNING WITH DIFFERENTIAL PRIVACY [73]

Abadi et al. [73] was the first to introduce a Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm. They use several techniques to prove their algorithm's security guarantee. They employ gradient clipping to bind the influence of a training example by some threshold C. They define this threshold C and noise scale per layer. Further, they employ an "accountant" to compute the privacy cost associated with accessing a data point. Finally, they employ a moments "accountant" who tracks a lower and upper bound for the privacy loss over the training process. They test their work on MNIST and CIFAR-10, where they achieve accuracies of 90% and 67%, respectively, under large noise levels. This work was foundational, as it was the first to apply DP to deep learning. However, model performance scales poorly under larger security guarantees because it is the first of its kind.

2) PHAN et al. 2017: ADAPTIVE LAPLACE MECHANISM: DIFFERENTIAL PRIVACY PRESERVATION IN DEEP LEARNING [74]

This work aims to intentionally add additional noise to features that are less relevant to the task. They use Layer-wise Relevance Propagation to determine feature relevance. This work uses LaPlace noise to achieve DP, adding additional noise to the non-relevant features. They further provide theoretical analysis and guarantee of their techniques and privacy. They test their framework on MNIST and CIFAR-10, where they achieve 90% and 85% accuracies, respectively. This work can increase model performance under stronger security constraints compared to [73]. This work requires large amounts of training to reach strong model performance.

3) LATENT: LOCAL DIFFERENTIAL PRIVACY FOR DEEP LEARNING [75]

This work proposes LATENT, a local differentially private training algorithm. LATENT allows data owners to add randomization layers before data leaves their device. This can be achieved by splitting the architecture of a CNN into a convolutional module, randomization module, and fully connected module. They show an accuracy of 91-96% on MNIST and CIFAR-10. The authors are the first to implement LDP, which reduces training time and computational complexity and increases accuracy/security. Additional communication is required to communicate with each client.

4) GONG et al. 2020: PRESERVING DIFFERENTIAL PRIVACY IN DEEP NEURAL NETWORKS WITH RELEVANCE-BASED ADAPTIVE NOISE IMPOSITION [76]

Gong et al. [76] present an Adaptive Differential Privacy-Preserving Learning framework, which performs feature relevance analysis to determine important features. Based on this relevance, additional noise is added to less relevant features, preserving better privacy. They derive mathematical proofs for leaked privacy bounds on gradient updates and even derive a differentially private Adam optimization algorithm for backpropagation. They improve on previous works by injecting the same amount of noise during each update instead of varying amounts. They test their experiment on Adult, MNIST, CIFAR-10, and DCCC, MIMIC-III, where they achieve 84%, 99%, 80%, 96%, and 94% accuracies, respectively. These techniques decrease the number of epochs needed to train each model but also come at the cost of reduced accuracy on more complex datasets.

5) DNN-DP (DIFFERENTIAL PRIVACY ENABLED DEEP NEURAL NETWORK LEARNING FRAMEWORK FOR SENSITIVE CROWDSOURCING DATA) [77]

Similar to [74] and [76], this work also performs feature relevance to inject additional noise into less important features through ϵ-DP, Global Sensitivity, and a Laplace Mechanism. Unlike other works, this work uses random forests to determine the importance of features. They further scale noise added to datasets based on an adaptive coefficient, which helps ensure more noise is added to less important features and less noise is added to important features. They provide a theoretical analysis of their technique. They test their framework on the US Census data set and obtain around 85% accuracy. This work shows stronger performance on the US Census data set than previous works. This work, however, only uses a single dataset and model type as a point of comparison.

6) BU et al. 2020: SCALABLE AND EFFICIENT TRAINING OF LARGE CONVOLUTIONAL NEURAL NETWORKS WITH DIFFERENTIAL PRIVACY [78]

The work done by Bu et al. [78] builds upon the ideas laid by [73]. They explore f-differential privacy (f-DP) which is a relaxation of (ϵ, δ)-DP. Using f-DP, they derive NoisySGD and NoisyAdam, which both make use of Poisson subsampling of data points, gradient clipping, and a Gaussian mechanism for gradient randomness. Compared to the work done by [73] and their moment's estimator, this work can achieve asymptotically smaller δ values for the same ϵ value.

TABLE 7. Summary of differential privacy frameworks. Root Mean Square Error (RMSE) Area Under the ROC Curve (AUC). Differential Privacy (DP). Local Differential Privacy (LDP). The "Non-Linear" column is omitted as all frameworks support arbitrary non-linear function evaluation. A scheme column is included in its place. Additionally, the *inference time* column is used as a *training epochs* column. All DP frameworks support training, inference, and operations while providing both model and data privacy.

Framework	Opensource	Threat Model	Scheme	Dataset	Training Epochs	Accuracy
Abadi et al. 2016 [73]	✓	Trusted Third Party	(ϵ, δ)-DP	MNIST, CIFAR-10	140, 125	95.00%, 70.00%
Phan et al. 2017 [74]	✓	Trusted Third Party	ϵ-DP	MNIST, CIFAR-10	500, 800	93.66%, 71.1%
LATENT [75]	×	Untrusted Third Party	ϵ-LDP	MNIST, CIFAR-10	12, 100	98.16%, 78.75%
Gong et al. 2020 [76]	×	Trusted Third Party	ϵ-DP	Adult Income, MNIST, CIFAR-10, DCCC, MIMIC-III	8, 8, 8 8	84.00%, 99.00%, 96.00%, 94.00%
DNN-DP [77]	×	Trusted Third Party	ϵ-DP	US Census Dataset	100	86.4%
Bu et al. 2020 [78]	✓	Trusted Third Party	f-DP	MNIST, Adult Income, IMDb, MovieLens	100, 18, 9, 20	98.0%, 84.0%, 83.8%, 0.915 RMSE
VPP [79]	×	Trusted Third Party	ϵ-DP	Purchase-100, Texas-100, CIFAR-10, CIFAR-100	×	81.60%, 51.02%, 85.13%, 75.20%
3A [80]	×	Trusted Third Party	ϵ-DP	Adult Income, MNIST, MIMIC-III	×	0.863 AUC, 0.818 AUC, 0.749 AUC,
Adamczewski et al. 2023 [81]	×	Trusted Third Party	(ϵ, δ)-DP	MNIST, CIFAR-10	×	98.09%, 81.09%

With NoisySGD, they achieve 98% and 84% accuracy on the MNIST and Adult Income datasets. Using NoisyAdam, they achieve 84.7% accuracy and 0.915 RMSE on the IMDB and MovieLens datasets. It shows the case of the largest models used so far in deep learning using DP. Their method only focuses on increasing the efficiency of DP without making improvements to data privacy.

7) VPP (PRIVACY PRESERVING MACHINE LEARNING VIA UNDERVOLTING) [79]

ML systems are susceptible to membership inference attacks, which leak private info from training data. This work proposes Privacy Preserving Volt, VPP, a lightweight inference-time approach to leveraging undervolting for PPML without the need to re-train the model. VPP injects computational randomness into a set of layers during inference through undervolting. Results show that this method improves utility by 32.93% lower than DP-SGD. Label-only attacks defeat related noise-based defenses, but VPP shows resilience. This work uses GPU acceleration to reduce the overhead of generating differentially private noise. They focus primarily on preventing membership inference attacks, in which some works outperform them under specific scenarios.

8) APPROXIMATE, ADAPT, ANONYMIZE (3A): A FRAMEWORK FOR PRIVACY-PRESERVING TRAINING DATA RELEASE FOR MACHINE LEARNING [80]

3A is designed to maximize the utility of ML with privacy using DP. They use Gaussian DP to anonymize a dataset, ensuring the resulting data is privacy-preserving and highly usable. They show their synthetic data and significant increases in classification performance compared to SOTA. However, this work sticks to simple datasets and models in their experimentation and evaluation.

9) ADAMZEWSKI et al. 2023: DIFFERENTIAL PRIVACY MEETS NEURAL NETWORK PRUNING [81]

This work studies the connection between DP and neural network pruning. They compare this work to differentially private sparse stochastic gradient descent (DP-SSGD), which consists of two modes for updating parameters. The first mode, called parameter freezing, freezes a set of weights that stay constant during the training. The second mode is parameter selection, in which a subset of parameters is chosen at each iteration to be updated. Further, they use the idea of private and public datasets to pre-train their models on the public dataset before using DP to train their models on private datasets. Their results show improvement over the DP-SGD algorithm, however, the requirement to pre-train on non-private datasets can be impractical in some cases.

Table 7 presents a detailed comparison of these frameworks. The features specific to differential privacy have been added to this table.

VII. TRUSTED EXECUTION ENVIRONMENTS

A trusted execution environment (TEE) is a secure area within a device consisting of hardware and an operating system. A TEE provides various features which depend on the individual implementations. Most commonly, a TEE isolates the processes from any process running in the standard OS of the device and from other processes inside the TEE. It also allows verification of the executed code and data [82]. There are five main types of TEEs today but Arm TrustZone and Intel SGX are the most common types used for machine learning training primarily because of their availability and stability. NVIDIA confidential computing (CCX) TEE works only with the NVIDIA GPUs. AMD SEV TEE is used to protects virtual machines thus not extensively used in ML. The Sanctuary TEE is an academic TEE designed from on ARM TrustZone ecosystem to support research. Figure 6 shows how a TEE can be used to carry out PPML inference on a general-purpose cloud setting with data transferred to it using cheap and fast symmetric key encryption such as Advanced Encryption Standard (AES).

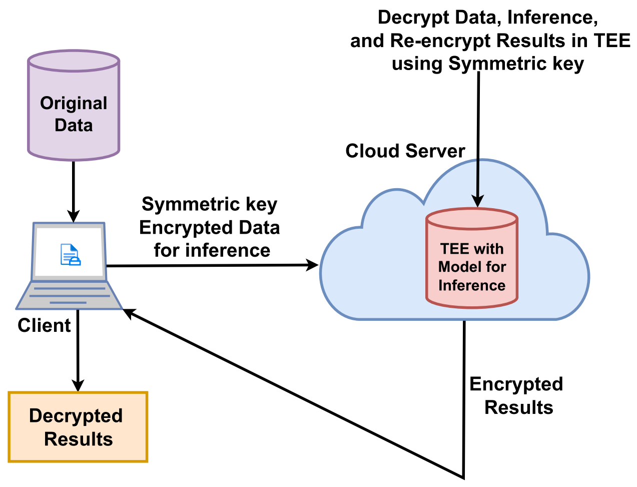

Decrypt Data, Inference, and Re-encrypt Results in TEE using Symmetric key
Original Data
Cloud Server
Symmetric key Encrypted Data for inference
TEE with Model for Inference
Client
Encrypted Results
Decrypted Results

FIGURE 6. Demonstration of a privacy-preserving machine learning using trusted execution environment.

A. FRAMEWORKS

1) PRABHU et al. 2020 [86]

Since edge devices such as sensors store raw data, they have become targets for attackers. Reference [86] proposed a method to train models on this data while preventing raw data leakages in the edge device. In their implementation, the applications cannot directly access the raw data inside the edge device as it is stored in a secure enclave provided by ARM TrustZone. The application can only get the result from the model running in the enclave. In their work, they abstract away the complexities of setting up ARM TrustZone and give developers flexibility. However, there is no hardware separation between sensors in the secure world and the normal world.

2) SECURE QNN [87]

To protect the privacy of a Quantized Neural Networks (QNN) on edge devices, [87] proposed a framework based on TrustZone. They assume that the service provider will provide a well-trained model to the client. The client is an honest but curious user and can access anything in the normal world. The model is a gray box in which part of it is in the normal world while part of the model is in the secure world. This framework evaluates the time an attacker needs to build a model from the disclosed information received from the target model. By using the evaluation, the framework splits the model into a normal world and a secure world. The layer needs less time to be reconstructed by the attacker, who will be in the secure world. The problem with this framework is that the splitting strategy of QNN is still very slow.

3) ORIGAMI [88]

Narra et al. [88] proposed a framework based on the Intel SGX by combining enclave execution, cryptographic blinding, and interspersed with accelerator-based computation. It runs the first few layers of the network in the enclave combined with cryptographic blinding, while the rest of the network directly runs in the GPU. The authors claim that an adversary cannot reproduce the data after the first few layers. It uses conditional Generative Adversarial Networks to infer the private data from the intermediate data. On the downside, the framework is vulnerable to side-channel attacks.

4) EnclaveTree [89]

It is a data stream training and inference framework based on Intel SGX that protects against access-pattern-based attacks on cloud providers. All models run in the enclave, and they use matrix multiplication to protect the access pattern in the enclave. This framework focuses on the Hoeffding Tree (HT) model and is 10x faster than the SOTA oblivious method used for binary feature classifications. However, attacks based on physical information, like rollback attacks and denial-of-service attacks, are possible.

5) SOTERIA [90]

It proposes a system based on Intel SGX to protect privacy in distributed ML. It offers a novel partitioning approach that allows specific operations to run outside the enclave using the GPU without leaking sensitive information. All work in the cluster runs in the enclave, assuming that users are semi-honest. It protects against adversarial attacks, model extraction attacks, model inversion attacks, reconstruction attacks, and membership inference. Compared with SOTA, this work reduces workload by up to 41%.

6) LEE et al. 2020 [91]

It is a framework proposed to protect Pytorch ML applications on the untrusted cloud from leaking input data and models without requiring any changes in these applications. All code in the cloud runs in the enclave of SGX. It uses Graphene Library OS to support the ELF file running on different platforms. The key generator runs on the machine that the user trusts. However, the framework offers no mitigations to hardware attacks.

7) OCCLUMENCY [83]

The Occlumency inference system is based on Intel SGX. It runs all the pipelines inside the SGX enclave. It proposes a memory-efficient inference to reduce the page swap in the memory and a partitioned convolution technique to accelerate the inference speed in the network. It uses a hash to check

TABLE 8. Summary of Trust Execution Environment (TEE). All frameworks support inference and traditional Non-linear operations. Visual Wake Words (VWW).

Framework	Opensource	ML Operations	Threat Model	Training	Dataset	Data Privacy	Model Privacy	Inference Time	Accuracy
Occlumency [83]	×	Conv, Linear	Malicious	×	ImageNet	✓	×	0.18s	×
AsymML [84]	×	Conv, Linear	Malicious	✓	CIFAR-10, ImageNet	✓	×	×, 9.8s	94%, 70.3%
Penetralium [85]	×	Conv, Linear	Malicious	×	MNIST, FMNIST, CIFAR-10, FaceScrub	×	✓	0.4, ×, ×	×, 91.16%, 63.5%, 70.0%
Prabhu et al. 2020 [86]	×	Cov, Linear	Malicious	×	Google Visual Wake Words	✓	✓	×	×
SecureQNN [87]	×	Conv, Linear	Semi-Honest	×	CIFAR-10, VWW	×	✓	0.829s, 0.625s	×
Origami [88]	✓	Conv, Linear	Malicious	×	ImageNet	✓	✓	0.06	×
EnclaveTree (HT) [89]	×	Hoeffding Tree	Semi-Honest	✓	Adult, REC, Covertype	✓	✓	19.05s, 128.28s, 2.36s	×
SOTERIA [90]	✓	ALS, GBT,PCA, Linear	Semi-Honest	✓	HiBench	✓	×	30.00s	×
Lee et al. 2020 [91]	✓	×	Malicious	✓	×	✓	×	×	×
DarKnight [92]	×	Conv, Linear	Malicious	✓	CIFAR-10, ImageNet	✓	×	×	87.50%, ×
Model Protection [93]	×	Conv, Linear	Semi-Honest	×	CIFAR-10, CIFAR-100, ImageNet	×	✓	×	93.3%, 72.1%, ×
Liu et al. 2022 [94]	×	Conv, Linear	Malicious	×	MNIST	✓	✓	45.674s	×

TABLE 9. TEE specific table. Membership Inference Attack (MIA), Gradient Inversion Attack(GIA), Model Inversion(MI), Reconstruction Attacks(RA), Side Channel Attacks (SCA).

Framework	Attack Protection	Integrity check	Acceleration	TEE hardware	Edge support
Occlumency [83]	MIA	×	×	SGX	✓
AsymML [84]	GIA, MIA	×	GPU	General	×
Penetralium [85]	MIA, MI	✓	GPU	SGX	✓
Prabhu et al. 2020 [86]	SCA	×	GPU		✓
SecureQNN [87]	MIA	×	×	TrustZone	✓
Origami [88]	MIA	×	GPU	SGX	×
Enclave-Tree (HT) [89]	SCA	×	×	SGX	×
SOTERIA [90]	MIA, MI, RA	✓	×	SGX	×
Lee et al. 2020 [91]	×	×	×	SGX	×
DarKnight [92]	MIA, SCA	✓	GPU	SGX	×
Model Protection [93]	×	×	GPU	SGX, TrustZone	✓
Liu et al. 2022 [94]	×	×	×	SGX	×

the integrity of the model thus prevent modifications in the untrusted memory. However, it only offers data privacy since they assume the model is open. Their system can not be used against side-channel, rollback, and DoS attacks. Their system runs on both Windows and Linux.

8) PENETRALIUM [85]

It is a system-algorithm co-design framework based on TEE. It focuses on ML inference on the edge device. It protects against membership inference and model inversion attacks. To increase inference performance, a novel computing engine designed for TEEs is proposed, and to increase privacy, it adds adaptive perturbation noise to the output confidence score. Compared with the TEE native method, it achieves the same accuracy and lower inference latency. Authors assume that the model runs on the edge provided by an honest third party who provides high-accuracy models, and they do not want their model's parameters to be stolen. They also assume that they will attack edge devices and that only the TEE part is trustworthy. On the downside, the framework does not protect against side-channel attacks.

9) SLALOM [95]

SLALOM is a PPML framework that uses Intel SGX and Sanctum TEEs to carry out PPML computations. It assigns the computation of all linear layers in a neural network to an untrusted third party and a co-located processor. It relies on a faster matrix multiplication algorithm proposed for TEEs for fast computation. The main advantage of SLALOM is fast verifiable inference. On the other hand, it is vulnerable to side-channel attacks present in SGX.

10) DarKnight [92]

It is a learning and inference framework that protects input privacy and computation integrity. By using TEE, this framework provides privacy and integrity verification. To accelerate the computation, the linear function is offloaded into a GPU. The authors claim an average improvement of 6.5x on different models compared with baseline model runs on SGX. The user provides the input data and model in this framework, which is trained and inferred on the cloud server. This framework encodes the data and model with a matrix between TEE, client, and GPUs, assuming the attackers are malicious on the cloud servers. To prevent injection faults from the GPUs, this framework checks the integrity of results obtained from the GPUs. However, it does not provide provable privacy guarantees against a scenario when all the GPUs participating in distributed training collude.

11) MODEL PROTECTION [93]

Hou et al. [93] propose a real-time inference system on the edge device. It is based on Intel SGX, though it is extensible to other TEEs like ARM TrustZone. It uses GPUs to accelerate the computation with the model provided by the service provider deployed on the user's edge device without leaking the model information to the user. The framework assumes that service providers are honest and will not record the user's data and inference results. The user is semi-honest and thus tries to learn information about models. To protect the model on the edge device, the framework adds random weights to the model, which makes the model output hardly recognizable and, thus, harder to retrain.

12) LIU et al. 2022 [94]

This framework proposes a solution of using cloud services to infer neural networks in a privacy-preserving setting. It is based on TEE and uses FHE technology to enable collaboration among mutually untrusted three parties (model provider, data provider, cloud server). In this framework, they assume that the data provider wants to use the model from the model provider to make the inference on the cloud, and the cloud server and model provider cannot know the result of the inference. They use the TEE to call the function to generate the encryption key of FHE and the FHE to encrypt the model and data that will be run in the Resource-rich Execution Environment (REE). The result of inference from REE will be encrypted by FHE and sent to the user from TEE. Compared with E2DM and CrytoNets, this work is more efficient.

Table 8 compares the frameworks based on the generic features used throughout this work, while Table 9 compares the different frameworks using specific TEE features.

VIII. HYBRID

In this section, we explore PPML frameworks that integrate techniques to enhance privacy and security in ML applications. They leverage the strengths of other techniques to mitigate the limitations of individual approaches. For instance, some frameworks combine MPC with HE, DP with FL, and MPC with TEEs. In Figure 7, we illustrate a practical scenario where an ML model is trained using Homomorphic Encryption to evaluate linear functions such as convolution and fully connected layers, followed by evaluation of non-linear functions such as ReLU within a TEE-like Arm TrustZone. This example is particularly significant and practical as many FHE schemes do not support the evaluation of non-linear functions, while TEEs typically face challenges with computational resources for large data evaluations thus the system leverages both techniques to provide both privacy and accuracy in the computation.

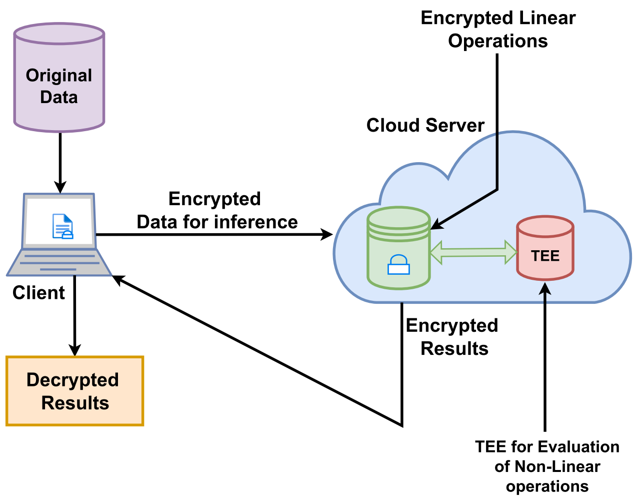

Original Data
Encrypted Linear Operations
Cloud Server
Encrypted Data for inference
TEE
Client
Encrypted Results
Decrypted Results
TEE for Evaluation of Non-Linear operations

FIGURE 7. Demonstration of a privacy-preserving deep learning training on encrypted data. Homomorphic encryption is used to evaluate linear functions while a TEE is used to evaluate non-linear functions.

A. FRAMEWORKS

1) HT2ML [96]

This framework integrates Intel SGX and HE to protect users' data and models. In this system, HE-friendly functions are performed outside the enclave, whereas HE-unfriendly non-linear functions are performed in the TEE. This framework implements an optimized matrix multiplication to accelerate HE computations. To prevent data leakages in the TEE, the framework provides data-oblivious execution blocks that leverage the oblivious primitives of the TEE. It demonstrates an $11\times$ faster HE computation on 6-dimensional data in linear regression. It shows a $196\times$ improvement in speed compared to SOTA HE CNN inference approaches. Though it shows better performance results compared to HE-only approaches and other works that combine TEE with HE, like TEEFHE [97] and HCNN [98], it significantly under-perform compared to pure TEE approaches.

2) RYFFEL AT AL. [99]

This framework introduces a generic PPML approach that combines FL, MPC, and DP while exposing a torch-like familiar deep learning API to the end-user. The framework focuses on providing premium ownership and secure data processing while introducing a valuable representation based on chains of commands and tensors. While the framework is very user-friendly, the accuracy can be as low as 67.1% on the MNIST dataset.

3) PySyft [100]

PySyft is a popular open-source PPML framework that provides secure and private data and model training through FL, DP, and encrypted computing through a numpy-like interface such as PyTorch, Keras, or TensorFlow integrated with deep learning, thus allowing users to integrate privacy-preserving techniques without modification of their systems.

It is already mainstream with support from major companies such as Microsoft, Google, Twitter, and Meta. PySyft provides abstractions for defining remote execution and secure aggregation of model updates in an FL setting. Customized accounts created on the system must be set up and used in applications.

4) PPML-TSA [101]

PPML-TSA is a versatile framework for privacy-preserving time series classification. It has been used with several models and datasets, including AlexNet, FCN, LeNet, and STM. It implements Deep learning methods in DP, FL, and HE out-of-the-box. The combination of DP and FL increases privacy by introducing errors, leading to minor accuracy loss. The framework is challenging to set up, making it hard to reproduce the results despite being open-source.

5) GenoPPML [102]

It is a domain-specific PPML framework developed for the context of sensitive genomic data processing. It combines MPC techniques and HE to train and infer with deep neural networks as well as other machine learning techniques. It was used to successfully solve breast Cancer prediction problems on gene expression datasets coming from different private sources while preserving privacy at the iDash competition. A 2-party logistic regression computation sees an 11x faster than other PPML techniques on the same CPU. Despite great performance on genomic datasets, the framework has not been tested with generic datasets like MNIST.

6) BOST ET AL. (MACHINE LEARNING CLASSIFICATION OVER ENCRYPTED DATA) [103]

It proposes a classification framework that provides privacy and security for different protocols, namely hyperplane decisions, Naive Bases, and Decision Trees. It proposes an interface with AdaBoost and combines HE, DP, and MPC. The protocol is very efficient and takes milliseconds to classify real-world patient data. The framework uses a library of building blocks for securely constructing flexible classifiers and shows the library's versatility by constructing a face detection classifier.

7) OBLIVIOUS MULTI-PARTY MACHINE LEARNING ON TRUSTED PROCESSORS (OMPML) [104]

OMPML is a framework that allows collaborative data analysis while preserving individual datasets. It additionally offers an interface to use trusted SGX processors for high performance through its support for TEEs. It supports tasks such as vector machines, matrix factorization, neural networks, decision trees, and k-means clustering. It uses a collaborative learning approach that provides prediction service in a cloud setting. It requires a code for the enclave to be public to ensure trust. While this framework supports different types of ML models and datasets, it is not open-source thus, not readily available.

8) GAZELLE [105]

The Gazelle framework is a low-latency and scalable private neural network inference framework. It offers privacy through a combination of HE with garbled circuits. It accelerates the convolution and matrix multiplication processes and automatically switches between HE and garbled circuits for processing encrypted data. To do this, it implements linear algebra kernels that map neural network layers to equivalent but optimized homomorphic matrix-vector multiplication and convolution routines. It designs and implements fast HE through the Gazelle library, which provides fast algorithms for basic homomorphic operations using SIMD. Though it offers a low-latency FHE approach, newer frameworks like [67] outperform it both in latency and accuracy.

9) PriModChain [106]

It is a framework that enforces privacy and trustworthiness on Internet of Things (IoT) data by combining DP, FL Ethereum blockchain, and smart contracts. It offers privacy, security, reliability, safety, and resilience in evaluated networks through programs developed in Python as sockets on a general-purpose computer. Though the paper uses the popular MNIST dataset to validate results, it states that more complex datasets demonstrate challenges in adopting it.

10) CHIRON [107]

Chiron is a PPML framework that combines FL and TEEs to provide privacy. It is implemented using SGX enclaves running standard ML training toolchain (including the popular Theano framework and C compiler) in the enclave. Still, the untrusted model-creation code from the service operator is further confined in a Ryoan sandbox to prevent it from leaking the training data outside the enclave. Chiron supports multiple enclave training by exchanging model parameters via a parameter server over different enclaves. It keeps both the data and models private from the users. The inference and training times of the framework are relatively high compared to their FHE counterparts, but it demonstrates great coordination between TEEs and FL.

11) PENTYALA et al. [108]

This work offers an MPC and DP model for PPML training and inference. An MPC protocol is used for model training and for perturbing the trained model coefficients with Laplace noise. The resulting MPC and DP approach achieves a higher accuracy compared to the pure DP approach while providing the same security guarantees. The work was also used at the IDASH2021 competition, and it demonstrated high accuracy and excellent performance on medical datasets. The main downside to this work is that it is not open-source and thus not easily reproducible.

12) AsymML [84]

AsymML is an asymmetric model decomposition framework based on TEE and DP. According to the low-rank features in

TABLE 10. Hybrid frameworks comparison.

Framework	Opensource	Operations	Threat Model	Non-Linear	Training	Inference	Dataset	Data Privacy	Model Privacy	Inference Time	Accuracy
HT2ML [96]	✓	Conv, Linear	Semi-Honest	Square	✓	✓	MNIST	✓	✓	1.39s	×
Ryffel et al. [99]	✓	Conv, Linear	Semi-Honest	Approximate	×	✓	Boston Housing, Pima Diabetes	×	✓	28.6 ms	67.1%
PySyft [100]	✓	Conv, Linear	Semi-Honest	approximate	✓	✓	Canonical Boston Housing	✓	✓	19.8ms	×
PPML-TSA [101]	✓	Conv, Linear	Semi-Honest	×	✓	✓	Entire UCR & UEA	✓	✓	×	×
GenoPPML [100]	×	Linear	Semi-Honest	×	×	✓	BC-TCGA, GSE2034, BC12-TCGA	✓	✓	2.50s, 2.38s, 2.50s	100%, 66%, 94%
Bost et al. [103]	×	Hyperplane Decision, Naive Bayes, Decision Tree	Semi-Honest	×	×	✓	real medical datasets (eg Breast Cancer)	✓	✓	204ms, 479 ms	×
Ohrimenko et al. [104]	×	Conv, Linear, Decision Tree, K-Means Clustering, SVM	Semi-Honest	ReLU	×	✓	UCIML Rep, MNIST, SUSY, MovieLens	✓	✓	×	×
Gazelle [105]	×	Conv	Semi-Honest	Square, ReLU	×	✓	MNIST, CIFAR20	✓	×	0.03s, 12.9s	×
PriModChain [106]	×	MCS System	Semi-Honest	×	×	✓	MNIST	✓	×	148s	×
Chiron [107]	-	Conv	Semi-Honest	ReLU	✓	✓	CIFAR-10, ImageNet	✓	✓	9.7h	89.56%
Pentyala et al. [108]	×	Linear	Dishonest-Majority	×	✓	✓	medical private dataset	✓	✓	0.09s	87.98%
AsymML [84]	×	Conv, Linear	Semi-Honest	ReLU	×	×	ImageNet, CIFAR-10	✓	✓	0.12s, 0.09s	×
SecureFL [109]	×	Conv, Linear	Semi-Honest	Square	✓	✓	MNIST	✓	✓	4260.3s	92.02%
SECO [110]	×	Conv, Linear	Semi-Honest	Approximate	×	✓	MNIST	✓	✓	11s	×

TABLE 11. Hybrid frameworks and protocols.

Framework	Techniques	Parties	Acceleration
HT2ML [96]	HE, TEE	×	GPU
Ryffel et al. [99]	MPC, DP	×	×
PySyft [100]	HE, MPC, DP, FL	×	×
PPML-TSA [101]	HE, DP, FL	×	×
GenoPPML [100]	HE, MPC	2	×
Bost et at. [103]	HE, MPC, DP	×	×
Ohrimenko et al. [104]	MPC, TEE	2	GPU
Gazelle [105]	HE, MPC	2	×
PriModChain [106]	DP, FL	×	×
Chiron [107]	TEE, FL	×	×
Pentyala et al. [108]	MPC, DP	2, 3, 4	×
AsymML [84]	DP, TEE	×	GPU
SecureFL [109]	TEE, FL	✓	GPU
SECO [110]	HE, MPC, FL	3	×

the dataset and intermediate features, the dataset can be split into low-rank data and residual parts, and the model can be split into a trusted and untrusted part. After applying DP to the residual dataset, it can be fed to the untrusted part, which runs on the GPU. The low-rank dataset is fed to the trusted part, which runs in the TEEs. It provides higher performance and the same privacy level as the DP-based method. This framework assumes that the attacker is malicious. They can access all hardware except the TEE environment and gain the dataset inside the untrusted environment. This framework can protect the model from model inversion attacks and gradient inversion attacks. However, this framework does not provide any protection against side-channel attacks or probing of physical devices.

13) SecureFL [109]

Kuznetsov et al. propose a framework leveraging SGX and TrustZone for FL. The cloud uses Intel SGX to protect the model, while on the edge device, they use the ARM Trustzone to protect the model [109]. In this framework, they assume that the attacker will act as a membership inference attack and that the attacker on the edge and cloud are both semi-honest. The attacker in the cloud has access to the system and all hardware on the cloud. Attackers on the edge device have access to the normal world. While this is a great approach, there are other common attacks in TEE, such as side channels not addressed by this work.

14) SECO [110]

It is a framework that offers private inference on multi-server hierarchy, giving nodes the ability to collaborate and compute predictions without compromising data security. It combines multiparty HE with garble circuits for model protection and is evaluated against MiniONN and ResNet-32. The framework offers an analysis of multi-input functional encryption using quadratic functional encryption to avoid information leakage in vertical FL. Though it shows great inference time and accuracy, there is a significant communication overhead.

Tables 10 show the comparisons between all hybrid frameworks while 11 shows the techniques, number of parties involved and acceleration status of the framework.

IX. GuardianML RECOMMENDATION SYSTEM

GuardianML is an extensible platform for selecting an effective PPML instance for a targeted use case application. It formulates the selection problem as a linear integer programming algorithm using different variables to calibrate the parameters of the PPML instance. For ease of use, it provides a web-based interface as an all-in-one recommender system for discovering the best-fitted framework for every use case. It offers six core functionalities: Search, Filter, Advanced Customization, Framework Inner Pages, and Addition of Frameworks.

A. HOW IT WORKS

Once this paper is accepted for publication, GuardianML will be made public online at https://prismo-ml.org. The main features include:

1) SEARCH

The platform's search feature is designed to be both comprehensive and user-friendly, offering a high degree of customization through six key attributes, each of which is optional, providing flexibility to the users. These attributes allow users to refine searches to meet specific needs and preferences. The features include PPML Technique, ML Models, Threat Model, Datasets, Training State, and Open-source State. GuardianML allows users to select from the main techniques of PPML discussed in this paper from Section III to Section VIII. PPML techniques helps narrow down search results to frameworks using specific PPML methodologies. ML models enable users to choose from the complete set of models available in the database. Multiple models can be selected, ensuring the search results include all relevant frameworks. The threat model is categorized into three types based on their protection guarantees against Malicious, Semi-honest, and Semi-Honest plus Trusted Third Parties. This classification covers the diverse security considerations across all PPML frameworks, allowing users to select the most relevant threat model for their use cases. Users can additionally select datasets that fit best with their use cases. The Training State allows users to specify their interest in privacy during different stages of the ML process, be it inference, training, or both. Lastly, users can choose whether they want open-source projects or not. Some ML frameworks analyzed are not open-sourced. For this reason, it helps identify frameworks that match the user's preferences for project accessibility and collaboration. By leveraging these features, users can conduct highly targeted searches, yielding relevant and valuable results tailored to their specific privacy, security, and ML requirements.

2) FILTERING

Filtering is a critical feature in GuardianML. It assist users in refining their search results to better align with their specific use case. It has four sets of filters used to refine search results:

- Hardware Acceleration: Computation cost is generally a significant challenge in ML. When combined with PPML techniques like HE, acceleration becomes crucial for training and inference performance. In PPML, it is typically achieved through GPU, ASIC, and FPGA accelerators. GPU acceleration leverages GPU features to speed up computation, ASIC acceleration employs specialized hardware to enhance speed, and FPGA acceleration uses digital design boards to outsource specific computational functions for better efficiency.
- Protocols: Different PPML techniques offer various protocols that are utilized to build applications. Users are allowed to filter results based on protocols for their selected technique. This filter is relevant for MPC, HE, and DP. This filter allows users to refine their search results according to the specific protocol requirements of their application, ensuring compatibility and alignment with their privacy needs. In DP, for example, local differential privacy and (ϵ, δ)-Differential Privacy lays the foundation for most frameworks.
- Libraries: They play a pivotal role in the development of applications utilizing HE and FL for PPML. Libraries offer diverse optimizations, protocols, and sometimes different security guarantees. When selecting an HE framework, a user's preference of the library can make a huge difference since different libraries offer different frontend techniques for integration. The HE libraries have different properties, implement different schemes, and perform differently [111].
- Specific Technique Features: Each PPML technique introduces critical considerations essential for secure computation. For example, in FL, factors such as the use of a centralized server and the support of edge devices can significantly influence a user's decision. Similarly, the number of parties supported by an MPC framework's methodologies.

B. RANKING

GuardianML's ability to rank potential frameworks based on user-defined importance and features is a crucial component of the recommendation engine. We use two approaches for ranking frameworks: the default and the user-optimization approaches. These approaches ensure that the system gives users the best-fitting solutions for all use cases.

1) DEFAULT APPROACH

This approach is used by default on search or filtered results obtained from the database to rank the frameworks before sending them to the frontend for display. Six main factors are used in ranking frameworks:

• Verifiable Results: We evaluate the performance of all PPML frameworks included in GuardianML. The frameworks that demonstrate the best performance are ranked highest in search results, followed by those with lower performance. Frameworks that we are unable to verify due to technical setup issues are ranked last. Performance is measured based on the accuracy obtained from the framework. Points are given to each framework by taking the ceiling of accuracy divided by 10 as shown in Equation 2 with 10 points being the maximum for accuracy greater than 90%.

$$p = \lceil accuracy/10 \rceil \tag{2}$$

As part of the reviewer's comment, the inference and training time is noted. It is very difficult to quantitatively compare the inference times since different systems implement different architectures, networks models, and run on different systems thus not considered in this work. After these evaluations and documentation, we create docker images with starter examples where necessary and shall make them available for users as an open-source project upon the publication of this work. A link to this Docker image shall be provided on the inner page of each verifiable framework. This is important for users interested in quickly prototyping and experimenting with PPML frameworks, as many of these projects are research-based and lack thorough documentation. Thus, setting up often requires technical skills that are not publicly discussed.

• Published Results: Another factor that influences our ranking algorithm is the published accuracy of the framework. Frameworks with higher inference accuracy are ranked higher in results, while those with lower accuracy are also ranked lower. Equation 2 is also used to give points to frameworks based on their published results. By factoring in Published results, we ensure that there is a balance between frameworks we cannot validate and those not made open-source.

• Open-source: The open-source state of a framework also influences our search results as open-source frameworks are generally more accessible and usable and offer developers flexibility and better community support. If the open-source field is not specified in the search, open-source frameworks generally appear higher in our rankings compared to closed-source projects. By prioritizing open-source frameworks, we aim to highlight options that foster collaboration and innovation. Thus, open-source frameworks are given 10 points, while closed-source frameworks are given 5 points.

• Model and Data Privacy: The primary objective of PPML is to provide model and data privacy in training and inference. Frameworks that provide both data and model privacy are considered more secure than those that provide just model or data privacy. For this reason, frameworks that offer both data and model privacy receive 10 points, while those that offer just data or model privacy receive 8 points.

• Threat Model Protection: The threat model protection of the framework is also an essential piece of privacy and security guarantees of the framework. The different threat model protections are awarded points as shown in Table 12. The points are given to frameworks based on the strength of the security guarantees offered through their threat model. For frameworks that offer different levels of protection, the highest level is considered during the evaluation.

TABLE 12. **Threat model protection points for framework.**

Threat Model Protection	Points
Protection against Malicious Adversaries	10
Protection against Semi-honest Adversaries	8
Use of Trusted Third-party	6

• Training and Inference Support: The scalability of PPML frameworks can be measured through the features supported. One way to quantify this is through their training and inference support. The ability to support both training and inference requires more work, thus resulting in better scalability. Frameworks that support both training and inference receive 10 points, while those that support only inference receive 8 points.

For all frameworks obtained in our search, the points assigned become weights of the framework to be used in ranking. Our methodology attempts to balance security, usability, performance, and accessibility to provide the users with an educated guess of the best-fitting framework. Linear Integer Programming (LIP) is used to achieve this objective by providing the user with personalized recommendations. A LIP optimization problem is formulated as follows:

$$\max \quad c^{\intercal}x \tag{3}$$
$$\text{Subject to } \; Ax \leq b. \tag{4}$$

In equations 3 and 4, given some $m, n \in \mathbb{N}$, $x \in \mathbb{Z}^n$ is the solution vector, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ are constraints, and $c \in \mathbb{R}^n$ are a set of weights [112].

We adopt the LIP setup to our setting, where x is a set of features such as verifiable results, published results, open-source state, model and data privacy, and threat model protection for each framework. c is a set of associated weights for each of the frameworks. The weights, c, are equal to the points discussed above, allowing all parameters of the frameworks to weigh equally in the ranking.

In GuardianML, the LIP problem boils down to an unconstrained maximization problem solved as shown in Equation 3 calculated for each framework. For our default case, the value of x for all frameworks is set to a median value of 5, customizable in the user-optimization approach. Let z_i be the sum of weights accumulated by framework i. f is the recommended framework, which is the framework with the maximum score obtained from the summation of points. As an exemplary use case, we assume we have frameworks denoted as z_1, z_2, and z_3. These frameworks have different threat models, semi-honest, malicious, and semi-honest, with a trusted third party. These frameworks have published accuracy of 92%, 97%, and 78% respectively. Also, z_1 is open-source, z_2 is not open-source, and z_3 is open-source. Additionally, z_1 and z_2 offer just inference, while z_3 offers both inference and training. To determine the framework that shall be recommended, let us construct the set of equations:

$$z_1 = 8x_1 + 10x_2 + 10x_3 + 8x_4 \tag{5}$$
$$z_2 = 10x_1 + 10x_2 + 5x_3 + 8x_4 \tag{6}$$
$$z_3 = 6x_1 + 8x_2 + 10x_3 + 10x_4 \tag{7}$$

Since our default value of x_i is 5, the sum for each framework is calculated, and frameworks are ranked as below:

$$z_1 = 8 * 5 + 10 * 5 + 10 * 5 + 8 * 5 = 180 \tag{8}$$
$$z_2 = 10 * 5 + 10 * 5 + 5 * 5 + 8 * 5 = 165 \tag{9}$$
$$z_3 = 6 * 5 + 8 * 5 + 10 * 5 + 10 * 5 = 170 \tag{10}$$
$$f = z_1 > z_3 > z_2 \tag{11}$$

2) USER-OPTIMIZATION RANKING APPROACH

The default ranking algorithm, described in the previous section, is employed when a user provides fundamental information, such as the desired PPML technique through search. In the default approach, the six factors of x contribute equally to the ranking algorithm. Additionally, the points attributed to the weights of x are generalized to make the most educated guess on frameworks. In the case the user provides additional information, we take a more sophisticated approach with ranking, giving them the option to customize all the parameters of the solution, which weights the final results in the LIP definition. We further divide the customization into two parts:

- **Factors:** In the default case described above, a default value of 5 is assigned to all x_i. This is generic as we assume that all the factors contribute equally to the final solution. GuardianML allows the user to calibrate the values of x between $[0, b]$, where b is chosen to normalize the range of weights per feature to allow fair comparisons. We set the value of $b = 10$ to give the user a reasonable range for calibration with equal positive and negative influence on factors. Thus, the median value of 5 is used for the default value of x_i. The value of x_i directly affects the impact that factor has on the weight of z_i in the maximization problem.
- **Points:** As discussed in the previous section, the points assigned to each framework factor become the weights of that framework in the LIP. In our default case, we scale these points to create the best-fitted guess for all use cases. In this advanced case, we allow the user to change the weights for every parameter considered. Weights are calibrated between $[0, d]$, and for the default case, d is 10 as shown above.

Let us add more constraints to the use case discussed in the default approach. The user is not particular about the threat model and sets its factor value to 3. The user is more interested in frameworks that use the ImageNet dataset. In most cases, the accuracy obtained with ImageNet ranges between 75% − 85%. Thus, the algorithm proposed in the default case will be biased on frameworks that evaluate this dataset only. For this use-case, the user changed the weighted value of accuracy to be 80% − 90% = 10, 70% − 80% = 10, and 90% − 100% = 5. Also, the user wants to pay more attention to approaches with better accuracy and changes its factor value to 8. Finally, the user does not care whether the framework is open-source, thus changing its factor value to 0. Re-calculating the sums of z_i in the equations above results in:

$$z_1 = 8 * 3 + 5 * 8 + 0 * 5 + 8 * 5 = 104 \tag{12}$$
$$z_2 = 10 * 3 + 5 * 8 + 0 * 5 + 10 * 5 = 120 \tag{13}$$
$$z_3 = 6 * 3 + 10 * 8 + 0 * 5 + 10 * 5 = 148 \tag{14}$$
$$f = z_3 > z_2 > z_1 \tag{15}$$

From the generalized case, the final LIP equations for GuardianML used to calculate z_i are written as:

$$z = \max\ [d^{\mathsf{T}} \odot c^{\mathsf{T}}]x_i \tag{16}$$

where $\odot$ represents the hadamard product. The pseudocode in Algorithm 1 shows how the recommendation engine of GuardianML works using customizable equations through scaling of features and parameter weights. This high level of customization ensures that the platform meets the diverse needs of its users, providing a robust tool for finding the most appropriate frameworks for every use case.

Algorithm 1 Default Ranking Algorithm

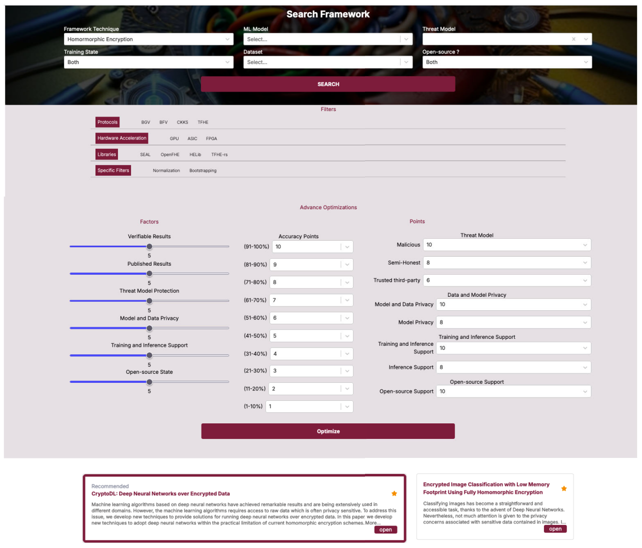
```
Require: Set of frameworks $\{x_i\}_{i=1}^{m}$. Weight vector $c$.
Ensure: $x_i \in \mathbb{Z}^n$, $c \in \mathbb{R}^n$ with $m, n \in \mathbb{N}$.
  $rankedFrameworks \leftarrow emptySortedList$
  for $j \in \{1, 2, \dots, m\}$ do
      $rank = c^{\mathsf{T}} x_j$
      $sortedInsert((x_j, rank), rankedFrameworks)$   ▷ Insert
          tuples of (framework, rank), sorted by rank.
  end for
  retur rankedFrameworks
```

Figure 8 shows the GuardianML search form, filters, and user optimizations in the ranking algorithm.

3) FRAMEWORK INNER PAGE

After searching and filtering results for a specific query, users can visit an inner page dedicated to every framework. This page provides a comprehensive summary of the framework, including essential information such as author details, an abstract of the publication, BibTeX citation for the associated publication, links to helpful information such as manuals, publications, and websites, details on the threat model, specifics on data and model privacy, training and inference support, open-source status, hardware acceleration, and technique specifics such as number of parties for MPC.

Additionally, the inner page features a special results summary format that consolidates findings from the original paper and the results from our verification process. This section includes details such as the dataset used, the ML



Search Framework

Framework Technique	Homormorphic Encryption
ML Model	Select...
Threat Model	
Training State	Both
Dataset	Select...
Open-source ?	Both

SEARCH

Filters

Protocols	BGV BFV CKKS TFHE
Hardware Acceleration	GPU ASIC FPGA
Libraries	SEAL OpenFHE HELib TFHE-rs
Specific Filters	Normalization Bootstrapping

Advance Optimizations

Factors

Verifiable Results	5
Published Results	5
Threat Model Protection	5
Model and Data Privacy	5
Training and Inference Support	5
Open-source State	5

Accuracy Points

(91-100%)	10
(81-90%)	9
(71-80%)	8
(61-70%)	7
(51-60%)	6
(41-50%)	5
(31-40%)	4
(21-30%)	3
(11-20%)	2
(1-10%)	1

Points

Threat Model	
Malicious	10
Semi-Honest	8
Trusted third-party	6
Data and Model Privacy	
Model and Data Privacy	10
Model Privacy	8
Training and Inference Support	
Training and Inference Support	10
Inference Support	8
Open-source Support	
Open-source Support	10

Optimize

Recommended
CryptoDL: Deep Neural Networks over Encrypted Data
Machine learning algorithms based on deep neural networks have achieved remarkable results and are being extensively used in different domains. However, the machine learning algorithms requires access to raw data which is often privacy sensitive. To address this issue, we develop new techniques to provide solutions for running deep neural networks over encrypted data. In this paper we develop new techniques to adopt deep neural networks within the practical limitation of current homomorphic encryption schemes. More...
open

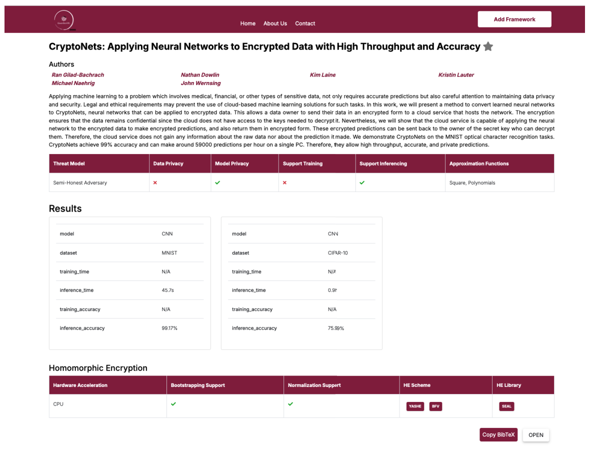
Encrypted Image Classification with Low Memory Footprint Using Fully Homomorphic Encryption
Classifying images has become a straightforward and accessible task, thanks to the advent of Deep Neural Networks. Nevertheless, not much attention is given to the privacy concerns associated with sensitive data contained in images. I...
open

FIGURE 8. **GuardianML advanced search frontend showing the search form, filters, and advanced optimizations. All this sections are used together in calibrating GuardianML to recommend the best-fitted framework for a specific use-case.**

model employed, training accuracy, inference accuracy, training time, and inference time. This detailed overview provided by GuardianML allows users to quickly assess whether the framework meets their requirements. The inner page also informs the user about the results we obtained while validating the framework. The summary of our work, the results, the link to the project, and instructions on setting it up. The results from our verification process come with comments on any additional feedback our team deems necessary. Importantly, users can download a ZIP file from this page containing the project's documentation and Docker image, which we prepared. This is essential for improving the adaptability and usability of open-source PPML frameworks. Since many of these frameworks stem from academic research, they often lack comprehensive documentation and may contain bugs, making setup challenging due to the specialized skills required, which are generally not readily available. Figure 9 shows an inner page of a framework on GuardianML. The example shown is for an unverified HE framework.

4) ADDING FRAMEWORK

In this work, we summarized the most relevant frameworks currently available in PPML. Looking forward, we anticipate the release of more frameworks, and to accommodate this development, this feature allows authors to submit their works to be added to GuardianML through a very comprehensive and easy-to-use form. This submission process is carried out through a user-friendly and detailed form, ensuring accessibility for all contributors. Each submission undergoes a thorough review by our team, where the framework's results and supporting information are carefully analyzed. Once approved, the framework is integrated into GuardianML and



CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy

Authors
Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Applying machine learning to a problem which involves medical, financial, or other types of sensitive data, not only requires accurate predictions but also careful attention to maintaining data privacy and security. Legal and ethical requirements may prevent the use of cloud-based machine learning solutions for such tasks. In this work, we will present a method to convert learned neural networks to CryptoNets, neural networks that can be applied to encrypted data. This allows a data owner to send their data in an encrypted form to a cloud service that hosts the network. The encryption ensures that the data remains confidential since the cloud does not have access to the keys needed to decrypt it. Nevertheless, we will show that the cloud service is capable of applying the neural network to the encrypted data to make encrypted predictions, and also return them in encrypted form. These encrypted predictions can be sent back to the owner of the secret key who can decrypt them. Therefore, the cloud service does not gain any information about the raw data nor about the prediction it made. We demonstrate CryptoNets on the MNIST optical character recognition tasks. CryptoNets achieve 99% accuracy and can make around 59000 predictions per hour on a single PC. Therefore, they allow high throughput, accurate, and private predictions.

Threat Model	Data Privacy	Model Privacy	Support Training	Support Inferencing	Approximation Functions
Semi-Honest Adversary	✗	✓	✗	✓	Square, Polynomials

Results

model	CNN
dataset	MNIST
training_time	N/A
inference_time	45.7s
training_accuracy	N/A
inference_accuracy	99.17%

model	CNN
dataset	CIFAR-10
training_time	N/A
inference_time	0.9h
training_accuracy	N/A
inference_accuracy	75.99%

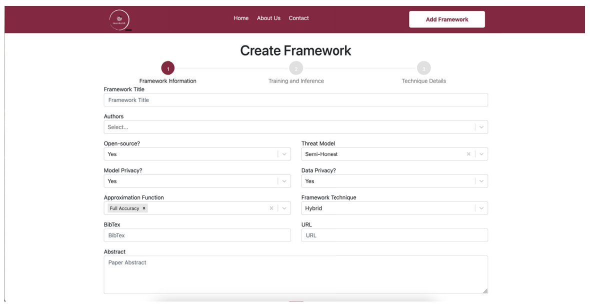
Homomorphic Encryption

Hardware Acceleration	Bootstrapping Support	Normalization Support	HE Scheme	HE Library
CPU	✓	✓	YASHE BFV	SEAL

Copy BibTeX OPEN

FIGURE 9. **An example of GuardianML Inner Page showing a framework's abstract, results and summarized information.**

incorporated into the recommendation engine. We consider this feature essential as it ensures every framework has an equitable opportunity to be part of the recommendation engine. This approach also very important as it helps to complete the system by addressing any potential omissions during this initial phase. This feature also plays a crucial role in GuardianML's ongoing development by identifying and addressing gaps or overlooked frameworks from the initial research phase. By facilitating seamless contributions and ensuring accurate, verified integration, it helps GuardianML stay up to date with the latest advancements in PPML, fostering a more comprehensive and dynamic resource for the entire research community. Figure 10 shows the first page of the form used in adding new frameworks into GuardianML.



Create Framework

1 Framework Information — 2 Training and Inference — 3 Technique Details

Framework Title	Framework Title
Authors	Select...
Open-source?	Yes
Threat Model	Semi-Honest
Model Privacy?	Yes
Data Privacy?	Yes
Approximation Function	Full Accuracy
Framework Technique	Hybrid
BibTex	BibTex
URL	URL
Abstract	Paper Abstract

FIGURE 10. **Adding a new framework into GuardianML.This feature allows extensibility and scalability of GuardianML.**

C. IMPLEMENTATION ARCHITECTURE

The system is built on a streamlined architecture consisting of three core components: the database, backend API, and frontend, as shown in Figure 11. Information submitted to the frontend is filtered, validated, and submitted to the backend for processing. The results are obtained from the database, processed, and sent to the frontend for display. In addition to the main features, GuardianML adds a data backup layer for protection, where a JSON file is generated for every framework inserted into the database and stored in an external system. These files are stored as backups and can be used at any time to recover the state of the system.

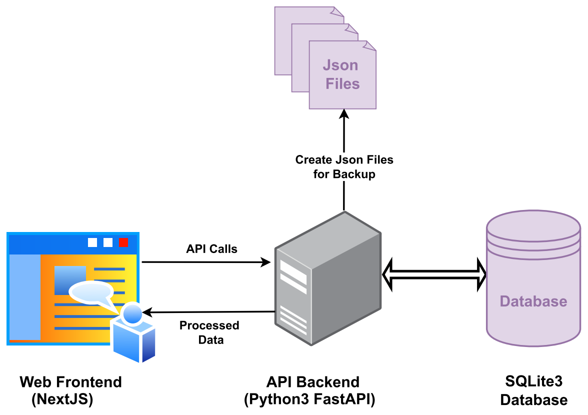


FIGURE 11. GuardianML ML architecture.

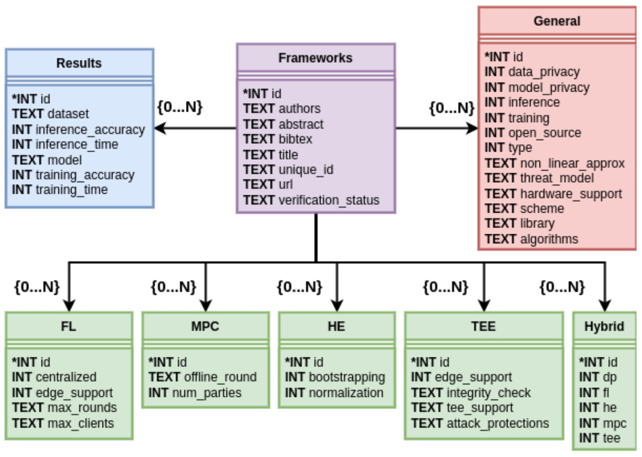


FIGURE 12. Simplified ER diagram of the database.

We use SQLite3 for the database as it offers the most recent and lightweight version of SQL, known for its efficiency and simplicity in managing data. Figure 12 visually represents the Entity-Relationship Diagram that outlines the database structure. We use Python FastAPI to develop the backend. This decision is rooted in its robust features that streamline API endpoints creation and integrate seamlessly with SwaggerHub for comprehensive API documentation. This choice ensures efficient API management and rapid deployment of functionalities, aligning perfectly with GuardianML's need for scalability and developer productivity. Next.js is used for the frontend as it simplifies the process of building complex user interfaces while effectively separating application states from views. Additionally, Next.js offers robust support for server-side rendering, enabling faster page loads and improved performance, which is crucial for delivering a responsive user experience.

X. USE CASES

We explore several use cases assessed using GuardianML and analyze the outcomes. To begin, we define the variables for our optimization problem as follows: Threat Model Protection (x_1), Model and Data Privacy (x_2), Published Accuracy (x_3), Verifiable Results (x_4), Open-source Status (x_5), and Training and Inference Support (x_6).

A. 1ST SCENARIO

We consider a scenario where a GuardianML user is looking for frameworks that support CNN inference, offer protection against semi-honest adversaries, and are open-source. Based on these criteria, GuardianML recommends the following top five frameworks: CryptoDL [63], LowMemory20 [67], CrypTFlow [42], PyHENet [64], and FLUTE [22]. In this default scenario, all x_i values are set to five, leading to the following equations and results for these frameworks, respectively:

$$8x_1 + 10x_2 + 10x_3 + 10x_4 + 10x_5 + 8x_6 = 280 \quad (17)$$
$$8x_1 + 10x_2 + 9x_3 + 10x_4 + 10x_5 + 8x_6 = 275 \quad (18)$$
$$10x_1 + 10x_2 + 5x_3 + 10x_4 + 10x_5 + 8x_6 = 265 \quad (19)$$
$$8x_1 + 10x_2 + 5x_3 + 10x_4 + 10x_5 + 10x_6 = 265 \quad (20)$$
$$10x_1 + 10x_2 + 7x_3 + 5x_4 + 10x_5 + 10x_6 = 260 \quad (21)$$

CryptoDL [63], represented by Equation 17, is an open-source HE framework that meets all the specified criteria. It offers protection against semi-honest adversaries and has been verified with accurate results that match those reported in the original publication. The framework achieved 99.56% accuracy on the MNIST dataset and 94.2% on the CIFAR-10 dataset, both of which are the highest reported for any HE framework on these datasets. This gives CryptoDL an advantage over LowMemory20, shown in Equation 18, which shares similar features but has a lower published accuracy of 85% on the CIFAR-10 dataset. While CrypTFlow provides protection against malicious adversaries, its published accuracy is significantly lower, with results of 76.47% and 74.25% on the two models. This highlights CryptoDL's superior performance over the other frameworks and explains why it is recommended.

B. 2ND SCENARIO

Assume the user places a high emphasis on the verifiable results of a framework, assigning it a factor weight of 10. They consider the published results less critical, giving them a factor weight of 2. The open-source nature of the framework is also not a top priority, with a weight of 3, while the protection against the threat model is considered highly important, given a factor weight of 8. Additionally, the user prefers deep networks tested on large datasets, understanding that these networks typically achieve accuracies ranging from 75-85%. As a result, they adjust the scoring so that accuracies in this range are awarded 10 points, while scores for 100% accuracy are reduced to 5 points. Consequently, the top recommended framework becomes CrypTFlow, followed by CryptoDL, PyHENet, LowMemory20, and PySyft. The equations below present the weightings for each framework:

$$10 * (8 + 5) + 5 * 2 + 10^2 + 10 * 3 + 8 * 5 = 310 \quad (22)$$
$$8^2 + 10 * 5 + 5 * 2 + 10^2 + 10 * 3 + 10 * 5 = 299 \quad (23)$$
$$8^2 + 10 * 5 + 10 * 2 + 10^2 + 10 * 3 + 8 * 5 = 294 \quad (24)$$
$$8^2 + 10 * 5 + 10 * 2 + 10^2 + 10 * 3 + 8 * 5 = 294 \quad (25)$$
$$8^2 + 10 * 5 + 0 * 2 + 10^2 + 10 * 3 + 10 * 5 = 294 \quad (26)$$

CrypTFlow is an MPC framework that defends against malicious adversaries by transforming TensorFlow code into secure inference for PPML code. It has been verified and tested on large datasets like CIFAR-10, using much deeper neural networks. The framework reports an accuracy of 93.23% on MNIST, and 76.47% and 74.25% on CIFAR-10 with deep networks such as ResNet50 and DenseNet121, respectively. No other framework has been evaluated with such deep networks in its published results, making CrypTFlow the best-suited option for this use case.

C. 3RD SCENARIO

Suppose the user opts to focus solely on an HE framework. They no longer consider the open-source nature of the framework as a priority, adjusting the points to 5 and the factor weight to 2. The user also decides that the verification of results is not important and changes the factor weight to 0. Additionally, the user selects a filter to only include frameworks that support the CKKS scheme. Based on these preferences, the recommended framework is PPDL [68], with E2DM [59], LowMemory20, PyHENet, and Lee et al. [65] rounding out the top five. The equations resulting from this query are as follows:

$$5 * (8 + 7) + 10 * 2 + 5 * 0 + 5 * 2 + 10 * 5 = 185 \quad (27)$$
$$5 * (8 + 7) + 10 * 5 + 5 * 0 + 5 * 2 + 8 * 5 = 175 \quad (28)$$
$$5 * (8 + 7) + 10 * 5 + 10 * 0 + 5 * 3 + 8 * 5 = 175 \quad (29)$$
$$5 * (8 + 7) + 5 * 5 + 10 * 0 + 5 * 2 + 10 * 5 = 160 \quad (30)$$
$$5 * (8 + 7) + 5 * 5 + 5 * 0 + 5 * 2 + 8 * 5 = 150 \quad (31)$$

In this scenario, PPDL [68] stands out as the recommended framework. It is validated exclusively on the MNIST dataset, achieving the highest accuracy of 97.84% among all HE frameworks. PPDL utilizes the CKKS scheme implementation from the HElib library to support both training and inference tasks. Given the user's priorities, the framework's published results are particularly significant in influencing their decision. Additionally, its capability to handle both training and inference gives it an edge over other frameworks. Based on the available research, PPDL is the most appropriate recommendation, offering strong inference accuracy and reliable support for both training and inference tasks. The analysis of these scenarios, supported by data from relevant research, illustrates that GuardianML successfully fulfills its intended purpose. It consistently identifies and recommends the most appropriate frameworks for each situation. The recommended frameworks demonstrate clear and significant advantages over alternatives, as shown by their performance and the supporting literature. This highlights GuardianML's ability to effectively match scenarios with the best-fitting frameworks, ensuring optimal results for each unique context.

XI. DISCUSSION

PPML has become an essential sub-field of ML and artificial intelligence. Though significant advancements have been made in recent years, more research efforts are needed to advance the field of PPML, especially with cryptographic primitives, as about 80% of frameworks using cryptographic primitives support just inference. Additionally, few PPML frameworks support large but well-used deep neural networks such as ResNet54, ResNet110, ResNet152, and VGG19. The application of PPML to sub-fields like reinforcement learning and spiking neural networks remains largely unexplored, with no significant frameworks addressing privacy issues in these areas. Furthermore, Verifiable PPML using cryptographic primitives such as zero-knowledge proofs and message authentication codes is still in the primitive stages, and very few works explore this concept for improved security and privacy in ML. Based on our analysis, Hybrid approaches seem to be the most promising approach for build the next generation of PPML frameworks. They take advantage of the different PPML techniques, reducing the communication and computation costs in MPC and HE, increasing the security guarantees in the case of FL and DP, and increasing computational efficiency in TEEs.

Unlike most previous research platforms in PPML which generally propose frameworks or compilers for specific instances, our research evaluates seventy PPML frameworks across different techniques, identifying common factors that can be extracted for comparison. We leverage these factors to develop a platform capable of conducting quantitative analysis and cross-technique framework comparisons in PPML. To the best of our knowledge, this is the first recommendation system designed for all PPML frameworks. GuardianML enables users to analyze and rank frameworks while providing a comprehensive overview, allowing them to quickly understand key features. Additionally, this work presents various open-source repositories containing well-documented resources and Docker images built from running versions of the frameworks. As a result, we offer a practical and structured resource for developers and researchers in the field of PPML.

XII. CONCLUSION

We have compiled essential techniques and frameworks for creating PPML applications. The frameworks reviewed demonstrate substantial progress in PPML, particularly in deep learning and classification. Additionally, we provide an open-source repository that includes frameworks with examples and setup instructions. The repository offers docker images for some frameworks accompanied by clear setup guidelines and basic examples. This resource is a foundational tool, enabling practitioners to effortlessly explore and adopt the latest advancements in PPML

Looking forward, our roadmap for GuardianML includes developing a sophisticated learning system that harnesses user interactions to dynamically assess framework and protocol relevance. This involves leveraging machine learning clustering algorithms to refine and personalize recommendations. Additionally, we envision integrating a robust review mechanism to augment our recommendation engine's efficacy, further enhancing user decision-making processes.

REFERENCES

[1] S. Shinde. (2022). *What Are Machine Learning Applications? Top 10 Industry and Real-world Use Cases*. Accessed: Mar. 30, 2024. [Online]. Available: https://emeritus.org/blog/machine-learning-what-are-machine-learning-applications

[2] S. Srivastava. (2023). *Future of Patient Care: The Use of Machine Learning in Healthcare*. Accessed: Mar. 30, 2024. [Online]. Available: https://appinventiv.com/blog/machine-learning-in-healthcare/

[3] R. Xu, N. Baracaldo, and J. Joshi, "Privacy-preserving machine learning: Methods, challenges and directions," 2021, *arXiv:2108.04417*.

[4] V. Lezginov. (2023). *Privacy-Preserving Machine Learning: Ml and Data Security*. Accessed: Mar. 30, 2024. [Online]. Available: https://scopicsoftware.com/blog/privacy-preserving-machine-learning/

[5] H. C. Tanuwidjaja, R. Choi, and K. Kim, "A survey on deep learning techniques for privacy-preserving," in *Machine Learning for Cyber Security*, X. Chen, X. Huang, and J. Zhang, Eds., Cham, Switzerland: Springer, 2019, pp. 29–46.

[6] S. Z. El Mestari, G. Lenzini, and H. Demirci, "Preserving data privacy in machine learning systems," *Comput. Secur.*, vol. 137, Feb. 2024, Art. no. 103605. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823005151

[7] A. Boulemtafes, A. Derhab, and Y. Challal, "A review of privacy-preserving techniques for deep learning," *Neurocomputing*, vol. 384, pp. 21–45, Apr. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219316431

[8] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Secur. Privacy*, vol. 17, no. 2, pp. 49–58, Mar. 2019.

[9] B. Pulido-Gaytán, A. Tchernykh, J. M. Cortés-Mendoza, M. Babenko, and G. Radchenko, "A survey on privacy-preserving machine learning with fully homomorphic encryption," in *High Performance Computing*, S. Nesmachnow, H. Castro, and A. Tchernykh, Eds., Cham, Switzerland: Springer, 2021, pp. 115–129.

[10] R. Podschwadt, D. Takabi, P. Hu, M. H. Rafiei, and Z. Cai, "A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption," *IEEE Access*, vol. 10, pp. 117477–117500, 2022.

[11] A. Bellini, E. Bellini, M. Bertini, D. Almhaithawi, and S. Cuomo, "Multi-party computation for privacy and security in machine learning: A practical review," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience (CSR)*, Jul. 2023, pp. 174–179.

[12] Q. Zhang, C. Xin, and H. Wu, "Privacy-preserving deep learning based on multiparty secure computation: A survey," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10412–10429, Jul. 2021.

[13] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, Jan. 2021, Art. no. 106775. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121000381

[14] M. Gong, Y. Xie, K. Pan, K. Feng, and A. K. Qin, "A survey on differentially private machine learning [review article]," *IEEE Comput. Intell. Mag.*, vol. 15, no. 2, pp. 49–64, May 2020.

[15] F. Mo, Z. Tarkhani, and H. Haddadi, "Machine learning with confidential computing: A systematization of knowledge," *ACM Comput. Surv.*, vol. 56, no. 11, pp. 1–40, Jun. 2024, doi: 10.1145/3670007.

[16] X. Li, B. Zhao, G. Yang, T. Xiang, J. Weng, and R. H. Deng, "A survey of secure computation using trusted execution environments," 2023, *arXiv:2302.12150*.

[17] M. Staňo and L. Hluchý, "A state-of-the-art survey on local training methods in federated learning," in *Proc. IEEE 23rd Int. Symp. Comput. Intell. Informat. (CINTI)*, Nov. 2023, pp. 89–92.

[18] M. Stano, L. Hluchý, M. Bobák, P. Krammer, and V. Tran, "Federated learning methods for analytics of big and sensitive distributed data and survey," in *Proc. IEEE 17th Int. Symp. Appl. Comput. Intell. Informat. (SACI)*, May 2023, pp. 705–710.

[19] I. Kholod, E. Yanaki, D. Fomichev, E. Shalugin, E. Novikova, E. Filippov, and M. Nordlund, "Open-source federated learning frameworks for IoT: A comparative review and analysis," *Sensors*, vol. 21, no. 1, p. 167, Dec. 2020.

[20] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang, "FATE: An industrial grade platform for collaborative learning with data protection," *J. Mach. Learn. Res.*, vol. 22, pp. 1–6, Jan. 2021.

[21] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N. D. Lane, "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.

[22] M. H. Garcia, A. Manoel, D. M. Diaz, F. Mireshghallah, R. Sim, and D. Dimitriadis, "Flute: A scalable, extensible framework for high-performance federated learning simulations," 2022, *arXiv:2203.13789*.

[23] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "FedML: A research library and benchmark for federated machine learning," 2020, *arXiv:2007.13518*.

[24] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konecný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," 2019, *arXiv:1812.01097*.

[25] H. Ludwig et al., "IBM federated learning: An enterprise framework white paper V0.1," 2020, *arXiv:2007.10987*.

[26] P. Foley, M. J. Sheller, B. Edwards, S. Pati, W. Riviera, M. Sharma, P. Narayana Moorthy, S.-H. Wang, J. Martin, P. Mirhaji, P. Shah, and S. Bakas, "OpenFL: The open federated learning library," *Phys. Med. Biol.*, vol. 67, no. 21, Oct. 2022, Art. no. 214001, doi: 10.1088/1361-6560/ac97d9.

[27] *Federated Learning for Healthcare Using NVIDIA Clara*. Accessed: Jul. 12, 2024. [Online]. Available: https://developer.download.nvidia.com/CLARA/Federated-Learning-Training-for-Healthcare-Using-NVIDIA-Clara.pdf

[28] P. Wang, C. Shen, H. R. Roth, D. Yang, D. Xu, M. Oda, K. Misawa, P.-T. Chen, K.-L. Liu, W.-C. Liao, W. Wang, and K. Mori, "Automated pancreas segmentation using multi-institutional collaborative deep learning," 2020, *arXiv:2009.13148*.

[29] M. N. Galtier and C. Marini, "Substra: A framework for privacy-preserving, traceable and collaborative machine learning," 2019, *arXiv:1910.11567*.

[30] F. Lai, Y. Dai, S. S. Singapuram, J. Liu, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "FedScale: Benchmarking model and system performance of federated learning at scale," 2021, *arXiv:2105.11367*.

[31] Y. Xie, Z. Wang, D. Gao, D. Chen, L. Yao, W. Kuang, Y. Li, B. Ding, and J. Zhou, "FederatedScope: A flexible federated learning platform for heterogeneity," 2022, *arXiv:2204.05011*.

[32] A. Wahrstätter, S. Khan, and D. Svetinovic, "OpenFL: A scalable and secure decentralized federated learning system on the Ethereum blockchain," *Internet Things*, vol. 26, Jul. 2024, Art. no. 101174. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S254266052400115X

[33] *NVIDIA Clara for Medical Devices*. Accessed: Dec. 7, 2024. [Online]. Available: https://www.nvidia.com/en-us/clara/medical-devices/

[34] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends Privacy Secur.*, vol. 2, nos. 2–3, pp. 70–246, 2018, doi: 10.1561/3300000019.

[35] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, Nov. 1982, pp. 160–164.

[36] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 19–38.

[37] J. Liu, M. Juuti, Y. Lu, and N. Asokan. (2017). *Oblivious Neural Network Predictions Via Minionn Transformations*. Cryptology ePrint Archive, Paper 2017/452. [Online]. Available: https://eprint.iacr.org/2017/452

[38] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 707–721.

[39] P. Mohassel and P. Rindal. (2018). *ABY3: A Mixed Protocol Framework for Machine Learning*. Cryptology ePrint Archive, Paper 2018/403. [Online]. Available: https://eprint.iacr.org/2018/403

[40] M. Byali, H. Chaudhari, A. Patra, and A. Suresh. (2019). *Flash: Fast and Robust Framework for Privacy-Preserving Machine Learning*. Cryptology ePrint Archive, Paper 2019/1365. [Online]. Available: https://eprint.iacr.org/2019/1365

[41] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "EzPC: Programmable and efficient secure two-party computation for machine learning," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroSP)*, Jun. 2019, pp. 496–511.

[42] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2020, pp. 336–353.

[43] A. Patra and A. Suresh. (2020). *Blaze: Blazing Fast Privacy-preserving Machine Learning*. Cryptology ePrint Archive, Paper 2020/042. [Online]. Available: https://eprint.iacr.org/2020/042

[44] N. Koti, A. Patra, R. Rachuri, and A. Suresh. (2021). *Tetrad: Actively Secure 4PC for Secure Training and Inference*. Cryptology ePrint Archive, Paper 2021/755. [Online]. Available: https://eprint.iacr.org/2021/755

[45] N. Koti, M. Pancholi, A. Patra, and A. Suresh, "SWIFT: Super-fast and robust privacy-preserving machine learning," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2651–2668.

[46] T. Lu, B. Zhang, L. Li, and K. Ren. (2023). *Aegis: A Lightning Fast Privacy-preserving Machine Learning Platform Against Malicious Adversaries*. Cryptology ePrint Archive, Paper 2023/1890. [Online]. Available: https://eprint.iacr.org/2023/1890

[47] J. Ma, Y. Zheng, J. Feng, D. Zhao, H. Wu, W. Fang, J. Tan, C. Yu, B. Zhang, and L. Wang, "SecretFlow-SPU: A performant and user-friendly framework for privacy-preserving machine learning," in *Proc. USENIX Annu. Tech. Conf.*, Boston, MA, USA, Jul. 2023, pp. 17–33. [Online]. Available: https://www.usenix.org/conference/atc23/presentation/ma

[48] E. F. Brickell and Y. Yacobi, "On privacy homomorphisms (extended abstract)," in *Proc. Workshop Theory Appl. Cryptographic Techniques*, 1988, pp. 117–125.

[49] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.

[50] J. H. Cheon, A. Kim, M. Kim, and Y. Song. (2016). *Homomorphic Encryption for Arithmetic of Approximate Numbers*. Cryptology ePrint Archive, Paper 2016/421. [Online]. Available: https://eprint.iacr.org/2016/421

[51] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, Jan. 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:1467571

[52] N. Aggarwal, C. Gupta, and I. Sharma, "Fully homomorphic symmetric scheme without bootstrapping," in *Proc. Int. Conf. Cloud Comput. Internet Things*, Dec. 2014, pp. 14–17.

[53] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. (2018). *TFHE: Fast Fully Homomorphic Encryption Over the Torus*. Cryptology ePrint Archive, Paper 2018/421. [Online]. Available: https://eprint.iacr.org/2018/421

[54] (Jan. 2023). *Microsoft SEAL (Release 4.1)*. Microsoft Research, Redmond, WA, USA. Accessed: Feb. 27, 2024. [Online]. Available: https://github.com/Microsoft/SEAL

[55] OpenFHE. *Openfhe on Github*. Accessed: Mar. 6, 2024. [Online]. Available: https://github.com/openfheorg/openfhe-development

[56] Zama. (2022). *TFHE-RS: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data*. [Online]. Available: https://github.com/zama-ai/tfhe-rs

[57] S. Halevi and V. Shoup. (2020). *Design and Implementation of Helib: A Homomorphic Encryption Library*. Cryptology ePrint Archive, Paper 2020/1481. [Online]. Available: https://eprint.iacr.org/2020/1481

[58] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, Mar. 2016, pp. 201–210.

[59] X. Jiang, M. Kim, K. Lauter, and Y. Song. (2018). *Secure Outsourced Matrix Computation and Application to Neural Networks*. Cryptology ePrint Archive, Paper 2018/1041. [Online]. Available: https://eprint.iacr.org/2018/1041

[60] A. A. Badawi, L. Hoang, C. F. Mun, K. Laine, and K. M. M. Aung, "PrivFT: Private and fast text classification with homomorphic encryption," *IEEE Access*, vol. 8, pp. 226544–226556, 2020.

[61] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "NGraph-HE: A graph compiler for deep learning on homomorphically encrypted data," in *Proc. 16th ACM Int. Conf. Comput. Frontiers*. New York, NY, USA: Association for Computing Machinery, Apr. 2019, pp. 3–13, doi: 10.1145/3310273.3323047.

[62] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, "CHET: An optimizing compiler for fully-homomorphic neural-network inferencing," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.* New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 142–156, doi: 10.1145/3314221.3314628.

[63] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep neural networks over encrypted data," 2017, *arXiv:1711.05189*.

[64] Q. Chen, L. Yao, Y. Wu, X. Wang, W. Zhang, Z. L. Jiang, Y. Liu, and M. Alazab, "PyHENet: A generic framework for privacy-preserving DL inference based on fully homomorphic encryption," in *Proc. 4th Int. Conf. Data Intell. Secur. (ICDIS)*, Aug. 2022, pp. 127–133.

[65] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30039–30054, 2022.

[66] H. Chen, R. Cammarota, F. Valencia, F. Regazzoni, and F. Koushanfar, "AHEC: End-to-end compiler framework for privacy-preserving machine learning acceleration," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[67] L. Rovida and A. Leporati. (2024). *Encrypted Image Classification With Low Memory Footprint Using Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2024/460. [Online]. Available: https://eprint.iacr.org/2024/460

[68] N. Jain, K. Nandakumar, N. Ratha, S. Pankanti, and U. Kumar, "PPDL—Privacy preserving deep learning using homomorphic encryption," in *Proc. 5th Joint Int. Conf. Data Sci. Manage. Data*. New York, NY, USA: Association for Computing Machinery, Jan. 2022, pp. 318–319, doi: 10.1145/3493700.3493760.

[69] A. A. Badawi, J. Chao, J. Lin, C. F. Mun, J. J. Sim, B. H. M. Tan, X. Nan, K. M. M. Aung, and V. R. Chandrasekhar. (2018). *Towards the AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data With GPUs*. Cryptology ePrint Archive, Paper 2018/1056. [Online]. Available: https://eprint.iacr.org/2018/1056

[70] A. Benamira, T. Guérand, T. Peyrin, and S. Saha, "TT-TFHE: A torus fully homomorphic encryption-friendly neural network architecture," 2023, *arXiv:2302.01584*.

[71] E. Chou, J. Beal, D. Lévy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster CryptoNets: Leveraging sparsity for real-world encrypted inference," 2018, *arXiv:1811.09953*.

[72] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2019, doi: 10.1137/1.9781611975949.

[73] M. Abadi, A. Chu, I. Goodfellow, B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. 23rd ACM Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[74] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive Laplace mechanism: Differential privacy preservation in deep learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 385–394.

[75] P. C. Mahawaga Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "Local differential privacy for deep learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5827–5842, Jul. 2020.

[76] M. Gong, K. Pan, Y. Xie, A. K. Qin, and Z. Tang, "Preserving differential privacy in deep neural networks with relevance-based adaptive noise imposition," *Neural Netw.*, vol. 125, pp. 131–141, May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608020300460

[77] Y. Wang, M. Gu, J. Ma, and Q. Jin, "DNN-DP: Differential privacy enabled deep neural network learning framework for sensitive crowdsourcing data," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 1, pp. 215–224, Feb. 2020.

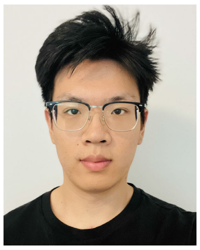
[78] Z. Bu, J. Dong, Q. Long, and W. Su, "Deep learning with Gaussian differential privacy," *Harvard Data Sci. Rev.*, vol. 2, no. 3, pp. 1–46, Sep. 2020. [Online]. Available: https://hdsr.mitpress.mit.edu/pub/u24wj42y

[79] M. S. Islam, B. Omidi, I. Alouani, and K. N. Khasawneh, "VPP: Privacy preserving machine learning via undervolting," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2023, pp. 315–325.

[80] T. Madl, W. Xu, O. Choudhury, and M. Howard, "Approximate, adapt, anonymize (3A): A framework for privacy preserving training data release for machine learning," 2023, *arXiv:2307.01875*.
[81] K. Adamczewski and M. Park, "Differential privacy meets neural network pruning," 2023, *arXiv:2303.04612*.
[82] N. Buchner, H. Kinkelin, and F. Rezabek, "Survey on trusted execution environments," *Network*, vol. 21, pp. 1–5, May 2022.
[83] T. Lee, Z. Lin, S. Pushp, C. Li, Y. Liu, Y. Lee, F. Xu, C. Xu, L. Zhang, and J. Song, "Occlumency: Privacy-preserving remote deep-learning inference using SGX," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, 2019, pp. 1–17.
[84] Y. Niu, R. E. Ali, and S. Avestimehr, "3LegRace: Privacy-preserving DNN training over TEEs and GPUs," 2021, *arXiv:2110.01229*.
[85] M. Yang, W. Yi, J. Wang, H. Hu, X. Xu, and Z. Li, "Penetralium: Privacy-preserving and memory-efficient neural network inference at the edge," *Future Gener. Comput. Syst.*, vol. 156, pp. 30–41, Jul. 2024.
[86] K. Prabhu, B. H. Jun, P. Hu, Z. Asgar, S. Katti, and P. Warden, "Privacy-preserving inference on the edge: Mitigating a new threat model," in *Proc. Res. Symp. Tiny Mach. Learn.*, 2020, pp. 1–9.
[87] M. Costa, T. Gomes, J. Cabral, J. Monteiro, A. Tavares, and S. Pinto, "SecureQNN: Introducing a privacy-preserving framework for QNNs at the deep edge," in *Proc. Int. Conf. Data Sci. Artif. Intell.* Singapore: Springer, 2023, pp. 3–17.
[88] K. Giri Narra, Z. Lin, Y. Wang, K. Balasubramaniam, and M. Annavaram, "Privacy-preserving inference in machine learning services using trusted execution environments," 2019, *arXiv:1912.03485*.
[89] Q. Wang, S. Cui, L. Zhou, O. Wu, Y. Zhu, and G. Russello, "EnclaveTree: Privacy-preserving data stream training and inference using TEE," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2022, pp. 741–755.
[90] C. Brito, P. G. Ferreira, B. Portela, R. Oliveira, and J. Paulo, "SOTERIA: Preserving privacy in distributed machine learning," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, 2023, pp. 135–142.
[91] D. Lee, D. Kuvaiskii, A. Vahldiek-Oberwagner, and M. Vij, "Privacy-preserving machine learning in untrusted clouds made simple," 2020, *arXiv:2009.04390*.
[92] H. Hashemi, Y. Wang, and M. Annavaram, "DarKnight: An accelerated framework for privacy and integrity preserving deep learning using trusted hardware," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 212–224.
[93] J. Hou, H. Liu, Y. Liu, Y. Wang, P.-J. Wan, and X.-Y. Li, "Model protection: Real-time privacy-preserving inference service for model privacy at the edge," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 4270–4284, Nov. 2022.
[94] P. Liu and W. Zhang, "Towards practical privacy-preserving solution for outsourced neural network inference," in *Proc. IEEE 15th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2022, pp. 357–362.
[95] F. Tramér and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," 2019, *arXiv:1806.0328*.
[96] Q. Wang, L. Zhou, J. Bai, Y. S. Koh, S. Cui, and G. Russello, "HT2ML: An efficient hybrid framework for privacy-preserving machine learning using HE and TEE," *Comput. Secur.*, vol. 135, Dec. 2023, Art. no. 103509. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823004194
[97] W. Wang, Y. Jiang, Q. Shen, W. Huang, H. Chen, S. Wang, X. Wang, H. Tang, K. Chen, K. Lauter, and D. Lin, "Toward scalable fully homomorphic encryption through light trusted computing assistance," 2019, *arXiv:1905.07766*.
[98] H. Xiao, Q. Zhang, Q. Pei, and W. Shi, "Privacy-preserving neural network inference framework via homomorphic encryption and SGX," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 751–761.
[99] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," 2018, *arXiv:1811.04017*.
[100] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, T. Ryffel, Z. N. Reza, and G. Kaissis, "PyAyft: A library for easy federated learning," in *Federated Learning Systems*. Cham, Switzerland: Springer, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236690571
[101] D. Mercier, A. Lucieri, M. Munir, A. Dengel, and S. Ahmed, "PPML-TSA: A modular privacy-preserving time series classification framework," *Softw. Impacts*, vol. 12, May 2022, Art. no. 100286. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665963822000355
[102] S. Carpov, N. Gama, M. Georgieva, and D. Jetchev. (2021). *GenoPPML—A Framework for Genomic Privacy-preserving Machine Learning*. Cryptology ePrint Archive, Paper 2021/733. [Online]. Available: https://eprint.iacr.org/2021/733
[103] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. (2014). *Machine Learning Classification Over Encrypted Data*. Cryptology ePrint Archive, Paper 2014/331. [Online]. Available: https://eprint.iacr.org/2014/331
[104] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th USENIX Secur. Symp.*, Austin, TX, USA, Aug. 2016, pp. 619–636. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/ohrimenko
[105] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. (2018). *Gazelle: A Low Latency Framework for Secure Neural Network Inference*. Cryptology ePrint Archive, Paper 2018/073. [Online]. Available: https://eprint.iacr.org/2018/073
[106] P. C. M. Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "A trustworthy privacy preserving framework for machine learning in industrial IoT systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6092–6102, Sep. 2020.
[107] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," 2018, *arXiv:1803.05961*.
[108] S. Pentyala, D. Railsback, R. Maia, R. Dowsley, D. Melanson, A. Nascimento, and M. D. Cock, "Training differentially private models with secure multiparty computation," 2022, *arXiv:2202.02625*.
[109] E. Kuznetsov, Y. Chen, and M. Zhao, "SecureFL: Privacy preserving federated learning with SGX and TrustZone," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Dec. 2021, pp. 55–67.
[110] S. Chen, "Distributed privacy-preserving machine learning via homomorphic encryption," Ph.D. dissertation, Dept. Electrical Comput. Eng., Univ. Toronto, Toronto, ON, USA, 2023.
[111] C. Gouert, D. Mouris, and N. G. Tsoutsos. (2022). *SoK: New Insights Into Fully Homomorphic Encryption Libraries Via Standardized Benchmarks*. Cryptology ePrint Archive, Paper 2022/425. [Online]. Available: https://eprint.iacr.org/2022/425
[112] K. Genova and V. Guliashki, "Linear integer programming methods and approaches-a survey," *Cybern. Inf. Technol.*, vol. 11, pp. 1–23, Jan. 2011.

NGES BRIAN NJUNGLE received the bachelor's degree in computer engineering from the University of Buea, Cameroon, in 2018, and the M.B.A. degree from the University of the People, Pasadena, California, in 2023. He is currently pursuing the Ph.D. degree in computer engineering with the Ira A. Fulton Schools of Engineering, Arizona State University. His research interests include the advancement and application of advanced cryptography, including homomorphic encryption, post-quantum cryptography, zero-knowledge proofs, formal methods, privacy-preserving applications, AI security, quantum computing, and next-generation secure systems.

ERIC JAHNS (Graduate Student Member, IEEE) received the bachelor's degree in computer science and mathematics from the University of Wisconsin–La Crosse, in 2023. He is currently pursuing the Ph.D. degree in computer science with Arizona State University. His primary research interests include neuromorphic computing, privacy-preserving machine learning, and interpretable/explainable ML/AI.



ZHENQI WU received the B.S. degree in electrical and automation engineering from Dalian Maritime University, in 2020, and the M.S. degree in electrical and computer engineering from the University of Florida, in 2022. He is currently pursuing the Ph.D. degree in computer science from Arizona State University. His research interests include neuromorphic computing and AI security.

LUIGI MASTROMAURO received the bachelor's degree in computer and electronics engineering and the master's degree in computer and automation engineering from the University of Ferrara, Italy, in 2019 and 2023, respectively. He is currently pursuing the Ph.D. degree in computer science with the Ira A. Fulton Schools of Engineering, Arizona State University. He is a Computer Engineer. His research focuses primarily on secure distributed systems and the next generation of secure quantum computing systems. He also actively researched secure distributed operating systems, attack detection, and measures to safeguard the integrity, confidentiality, and availability of computer systems in a decentralized application employing various techniques, including secure and private machine learning methods and fairness and robustness in federated learning.

MILAN STOJKOV received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the Department of Computing and Control Engineering, Faculty of Technical Sciences, University of Novi Sad, in 2014, 2015, and 2022, respectively. He has been an Assistant Professor with the Department of Computing and Control Engineering, Faculty of Technical Sciences, University of Novi Sad, since 2023. His primary research interests include information security, critical infrastructures, secure distributed systems, edge computing, and software architectures.

MICHEL A. KINSY (Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), in 2013. He is the Founding Director of the ASU Secure, Trusted, and Assured Microelectronics (STAM) Center. He currently holds the position of an Associate Professor with the Ira A. Fulton Schools of Engineering, Arizona State University. Prior to joining the ASU Faculty, he was an Associate Professor with the Department of Electrical and Computer Engineering, Texas A&M University (TAMU), where he also was the Associate Director of the Texas A&M Cybersecurity Center (TAMC2). He also held faculty positions with Boston University and the University of Oregon. From 2013 to 2014, he was a fully cleared member of the Technical Staff with the MIT Lincoln Laboratory. He focuses his research on secure computer architecture, hardware-level security, and efficient hardware design and implementation of post-quantum cryptography systems. He is an MIT Presidential Fellow and a recipient of the Inaugural Skip Ellis Career Award.