

# SoK: Wildest Dreams: Reproducible Research in Privacy-preserving Neural Network Training

Tanveer Khan  
Tampere University  
Tampere, Finland  
tanveer.khan@tuni.fi

Mindaugas Budzys  
Tampere University  
Tampere, Finland  
mindaugas.budzys@tuni.fi

Khoa Nguyen  
Tampere University  
Tampere, Finland  
khoa.nguyen@tuni.fi

Antonios Michalas\*  
Tampere University  
Tampere, Finland  
antonios.michalas@tuni.fi

## ABSTRACT

Machine Learning (ML), addresses a multitude of complex issues in multiple disciplines, including social sciences, finance, and medical research. ML models require substantial computing power and are only as powerful as the data utilized. Due to the high computational cost of ML methods, data scientists frequently use Machine Learning-as-a-Service (MLaaS) to outsource computation to external servers. However, when working with private information, like financial data or health records, outsourcing the computation might result in privacy issues. Recent advances in Privacy-Preserving Techniques (PPTs) have enabled ML training and inference over protected data through the use of Privacy-Preserving Machine Learning (PPML). However, these techniques are still at a preliminary stage and their application in real-world situations is demanding. In order to comprehend the discrepancy between theoretical research suggestions and actual applications, this work examines the past and present of PPML, focusing on Homomorphic Encryption (HE) and Secure Multi-party Computation (SMPC) applied to ML. This work primarily focuses on the ML model’s training phase, where maintaining user data privacy is of utmost importance. We provide a solid theoretical background that eases the understanding of current approaches and their limitations. We also provide some preliminaries of SMPC, HE, and ML. In addition, we present a systemization of knowledge of the most recent PPML frameworks for model training and provide a comprehensive comparison in terms of the unique properties and performances on standard benchmarks. Also, we reproduce the results for some of the surveyed papers and examine at what level existing works in the field provide support for open science. We believe our work serves as a valuable contribution by raising awareness about the current gap between theoretical advancements and real-world applications in PPML, specifically regarding open-source availability, reproducibility, and usability.

## KEYWORDS

Homomorphic Encryption, Multi-party Computation, Neural Networks, Privacy-Preserving Machine Learning

\*Also with RISE Research Institutes of Sweden.

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit <https://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.  
Proceedings on Privacy Enhancing Technologies YYYY(X), 1–21  
© YYYY Copyright held by the owner/author(s).  
<https://doi.org/XXXXXXXX.XXXXXXX>



## 1 INTRODUCTION

Due to scientific advancements, currently Machine Learning (ML) is widely used in a variety of applications such as image classification, stock predictions, machine translation, and cancer cell detection to name but a few. The benefits of ML comes at a cost of having significant computational overhead, which leads data scientists to turn to Machine Learning-as-a-Service (MLaaS) in order to outsource computations. However, the use of MLaaS has raised significant security and privacy concerns. More precisely, a plethora of works shows how to successfully attack ML algorithms and gain insight into private data. Examples of such attacks are reconstruction attacks [163], model inversion attacks [71], and membership inference attacks [134]. As far as privacy is concerned, ML algorithms are used in fields such as healthcare, personalization and virtual assistants, as they can easily leak sensitive information related to the users. In this survey, we focus on covering existing works, which aim to preserve data privacy in ML algorithms.

Privacy-Preserving Machine Learning (PPML) is a research field focused on enhancing data privacy in ML using Privacy-Preserving Techniques (PPTs). These techniques consist of cryptographic, distributed, hybrid, and data modification approaches (Figure 1). PPML is a highly active line of research, with a significant amount of work in literature [15, 35, 36, 70, 103, 140]. The main techniques used to achieve PPML are: Homomorphic Encryption (HE) [59], Secure Multi-party Computation (SMPC) [155], Federated Learning (FL) [143], Differential Privacy (DP) [51] and Functional Encryption (FE) [27]. The main aim of this Systemization of Knowledge (SoK) paper is to survey and compare State-of-the-Art (SoTA) works in the area of HE and SMPC-based PPML in the training phase.

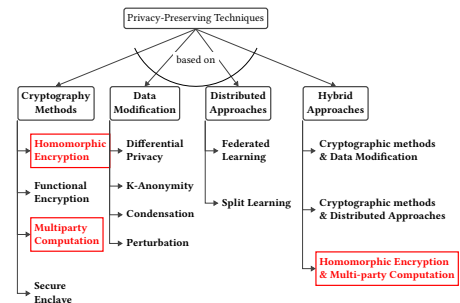


Figure 1: Privacy-preserving Techniques. This SoK covers the privacy-preserving techniques marked in red.

HE schemes, allow involved parties (users or service providers) to perform computations such as addition, multiplication or bitwise operations on encrypted data. After the computation, the format

of the input data remains intact, but its value changes [16]. This property of “privacy homomorphism” was initially theorized by Rivest, Adleman, and Dertouzos [124]. The researchers believed that, if the input data was homomorphic, they would be able to perform computations on encrypted data, similar to how multiplication is performed by the Rivest-Shamir-Adleman (RSA) public-key cryptosystem. As a result, first generation HE schemes, like Paillier [113] were only able to homomorphically compute a single type of mathematical operation. This changed in 2009, when C. Gentry [59] proposed the first fully HE scheme. Said scheme would allow multiple operations on encrypted data. However, because of the computational complexity, various researchers, including Gentry himself, began developing leveled HE schemes [33] or somewhat HE schemes [55, 119] to limit the amount of computations per system through different means.

These developments of HE have allowed users to store fully encrypted data in a cloud, while retaining their ability to perform computations on the encrypted data. This minimizes the risks of a malicious service provider or employee eavesdropping on confidential information, that is being stored in their service [76]. A general example of HE would be, if an encryption scheme would take two separate encrypted inputs  $E(a)$  and  $E(b)$ , where  $E(\cdot)$  is an encryption function, and by using an addition or multiplication algorithm it would be able to compute  $E(a) + E(b) = E(a + b)$  or  $E(a) \times E(b) = E(a \times b)$  respectively, without having to decrypt the inputs [59]. Aside from being used in cloud services [120] to protect data, HE can be applied to scenarios such as keeping medical and genome [151] information private in ML applications.

SMPC aims to create methods that enable different parties to jointly compute a function on their private data, while preserving privacy [154]. SMPC was introduced as a secure two-party computation (2PC) by Andrew Yao [155] and was generalized into SMPC by Goldreich, Micali, and Wigderson [64]. In 2PC settings [154], the two parties jointly compute a function on their inputs without disclosing the values of their inputs to the other party. The two-party setting can be extended to three (3PC) [23, 24, 102], four (4PC) [34, 65, 74], or, more generally,  $n$ -party [96, 99, 162] settings, where these parties collaborate to privately compute a joint function of their inputs.

Said feature has great potential, as it can be applied in any situation where sensitive data from two or more parties are computed. Additionally, it could help with many ML applications, currently infeasible due to privacy concerns. An application example is the simplified version of training ML models on private datasets held by different parties and evaluating one party’s private model using another party’s private data. Other applications for SMPC are private DNA comparison, privacy-preserving auctions, and more. To date, SMPC has made substantial success in several real-world situations, with a significant payoff to society. For instance, it has been used to securely analyze the wage data of 112,600 employees in the greater Boston area in order to quantify pay disparities by gender and race [144]. SMPC has also been used to allow satellite operators to calculate the likelihood of their satellites colliding without having to share their underlying private orbital information [68].

## 1.1 Motivation & Contribution

**Motivation:** ML allows interested parties (e.g. companies, researchers, etc.) to train models either locally or in an interactive distributed environment. Privacy is guaranteed in the scenario where a model is trained on a local machine, as in this case, no models or data are shared. However, there are concerns regarding the level of security when deploying ML in a distributed environment. For example, in a scenario where parties share their data to train a model. To address privacy concerns, various PPML techniques have been proposed. These PPML techniques take both ethical and legal concerns into account. Specifically, in privacy-preserving training, data privacy is demanded<sup>1</sup>. In privacy-preserving MLaaS, model privacy is also necessary besides data privacy so that neither the client nor any other parties learn anything about the model parameters on cloud servers, other than what can be learned from the final prediction.

The main motivation behind our study is to better understand the landscape of MLaaS in a data-sensitive context by employing PPML techniques. In this study, we focus primarily on the two cryptographic techniques: HE and SMPC – that are employed in PPML training. The other methods are also employed for PPML in various ways, but, here, they are not taken into consideration. This decision is backed by a number of reasons some of which are covered here. Secure Enclaves (SE) [129] is a hardware security approach and requires specialized hardware for computation, while HE and SMPC are pure cryptographic approaches and do not require such hardware for computation. As our survey exclusively examines application and non-hardware-based approaches, SE is not included in this survey. The second cryptographic method, FE, has a very constrained range of applications. It has numerous effective constructs for evaluating linear and quadratic functions, but only a few known FE schemes can be used to evaluate generic functions effectively. Due to the limited number of applications and existing studies, FE does not fall within the scope of this work. Furthermore, HE and SMPC are explicitly optimized for a specific notion of privacy while other approaches, such as distributed approaches, do help enhance privacy, but may not inherently protect privacy and are known to have privacy vulnerabilities [15, 115]. Hence, they do not form part of the scope of this work. Accordingly, while data modification, such as DP, is a common approach of PPML, it too is beyond the purview of our study, since the techniques we address utilize cryptography to hide information, while enabling computations over it. Such techniques also suffer from reducing model utility and are best used in hybrid settings.

While it is understood that not all research could offer open-source implementations (OSI) that is easy to install and run due to various constraints, doing so would tremendously enhance the value of a published paper. As analyzed in [111] several works in the secure ML domain do not provide any form of OSI or the provided code does not reproduce the expected results<sup>2</sup>. So, another key motivation for our work is to examine whether existing works in PPML

<sup>1</sup>Neither the cloud server nor any other involved parties, learn anything about out-sourced data of clients other than their size.

<sup>2</sup>[111] covers a wide range of ML papers published in top security venues and studies the reproducibility of those papers. The article analyses a variety of security related topics in ML, but does not go in to detail of the techniques covered or the results of the covered articles. Our paper analyses the reproducibility of the articles using HE and SMPC in PPML training and analyses their novelty, advantages and limitations.

provide support for open science by allowing others to successfully reproduce their results and use the designed models. For this, we focus on evaluating systems with OSI and compare our results to those reported in the original papers. Furthermore, we base our evaluation on another aspect: the threat model. As our study is centered around two PPTs, namely HE and SMPC, which primarily operate under semi-honest threat model, this choice provides a meaningful framework for assessing these works.

**Contribution:** In this context cover preliminaries for both HE and SMPC techniques, as well as examine and compare the libraries used to implement various schemes. We chose 9 SotA papers on HE and 17 on SMPC implementation in PPML, compared the methodologies and evaluated the outcomes. More specifically:

- We introduce the intersection of both ML and privacy fields placing special emphasis on cryptographic techniques used to protect the data.
- We lay the basic but substantial theoretical foundation that helps researchers comprehend current HE and SMPC-based methods to PPML. We also go through basic ML fundamentals and describe often-used datasets.
- We thoroughly review of HE and SMPC-based PPML literature, emphasizing on the strengths and shortcomings of various approaches and assessing how they supplement one another.
- We analyze various aspects of the implementations such as security, computational complexity, and adversarial models to get an overview of the current SotA implementations. We examine the limitations that prevent the existing HE and SMPC-based PPML solutions from being implemented into real-world settings, mostly due to issues with efficiency and usability.
- We also highlight the importance of OSI. By encouraging researchers to prioritize OSI, we aim to bridge the gap between theoretical advancements and real-world applications. This promotes improved reproducibility, wider adoption and impact, and long-term sustainability in the field of PPML.
- We lay out future research directions aimed at improving existing works in terms of performance and security.

**Comparison to related surveys:** Although some similarities are inevitable, our work differs from similar works [35, 92, 108, 114] in many aspects. In this article, we overview the entire spectrum of PPML and narrow down our analysis to HE and SMPC describing their functionality and limitations in depth. We mainly focus on the training part of DL, where user privacy is of critical concern. To further differentiate our work, we cover the functionality and performance of SotA PPML approaches and attempt to reproduce the results of OSIs. Aside from that, we compare the frameworks covered in terms of security, availability and performance and provide insight into available libraries. We believe that these two last tasks constitute the main contribution of this work. This is because these tasks aim to support open science and reproducible research and can give valuable insights to researchers using these libraries to conduct further research or to build modern privacy-preserving online services. Since our work is closely related to the work of [35] and [108], we provide a detailed comparison between our work, [35] and [108] in Table 1.

## 1.2 Organization

The rest of paper is organized as follows. In section 2, we define the scope and methodology. We overview the primary SotA HE and SMPC implementations of the SoK in section 3, more specifically we talk about how they are implemented, what datasets and which schemes are used. This section also compares the documented results of the research papers with other implementations mentioned in the original paper and provides an overview of the results to compare each implementation to one another. Then section 4 covers the experiments we conducted for both HE and SMPC implementations and the results we received from our tests. This section also covers various aspects of the HE and SMPC protocols such as the libraries used and how they differ, as well as the security, computational complexity and adversarial model of the proposed HE and SMPC implementations. The main takeaways from the survey are provided in section 5, followed by challenges and future directions in section 6, and conclude the paper in section 7. In Appendix A, we overview the categorization, history and schemes of HE and SMPC and also the formal definitions for both. We also refer to some of the libraries used to realize these schemes into various implementations for secure and private storage and computations.

## 2 SCOPE AND METHODOLOGY

Cryptographic techniques encounter difficulties when used with ML. For example, although HE can relieve the client endpoint of a significant workload, it can only compute a limited number of operations, when dealing with complex problems (due to performance issues). In addition, using HE for ML can be complicated, as some ML operations, such as non-linear activation functions, are incompatible with most HE, except TFHE [42], and can be simulated with polynomial approximation. As such, the main concerns when dealing with PPML using HE, relate to efficiency and usability. PPML can also be deployed in a collaborative setting using SMPC, where different stakeholders contribute with their data to a common goal. The main concern using SMPC is the possibility of malicious behavior by an external entity or even by a subset of participating parties as well as the high communication overhead.

It is important to research and assess various aspects of both HE and SMPC to better understand their effectiveness and usability, as both of the selected approaches have different advantages and disadvantages. The paper studies various aspects of each of the proposed approaches, such as the threat model, the supported layers, the corresponding techniques, and the evaluation datasets. As for SMPC, we cover the networking type and the NN architecture adopted in said approaches to better evaluate the testing environments and results. For HE, we review the HE setting to gain knowledge on current protocol design trends and the differences that occur depending on the setting chosen. To approach these aspects we overview relevant literature and perform experiments on available OSIs of the selected approaches. Each aspect contains information that helps analyze the proposed approaches in terms of both applicability and security. Analyzing the threat model allows us to gain a better understanding of the attacker’s abilities when faced with the protocol. This in turn describes the potential security of the protocol. Aspects such as the supported layers and evaluation datasets show the applicability of the protocol

**Table 1: Comparison between our paper, [35] and [108]**

|                                  | [35]   | [108]   | Our Paper   |
|----------------------------------|--|---|---|
| Focus                            | Study the landscape of PPML in data sensitive contexts through HE, SMPC and Hybrid Techniques  | Study 53 papers on privacy-preserving neural network based on HE and SMPC   | Study the landscape of PPML training as a service using HE and SMPC and Hybrid Techniques   |
| Methodology                      | <ul style="list-style-type: none"> <li>• Problem addressed, training or inference</li> <li>• The architecture proposed, i.e., centralized, distributed, or hybrid</li> <li>• Privacy goals and adversarial model</li> <li>• The particular techniques involved, i.e., SMPC, HE and/or others</li> <li>• The issues considered regarding efficiency and usability.</li> </ul> | There is not a clear framework on how the paper studies each work, but rather it systematizes notable works based on their use of cryptographic primitives and tricks, hence point out the relationships between these works  | <ul style="list-style-type: none"> <li>• Threat model, supported layers and evaluation datasets</li> <li>• Cryptographic technique used (e.g. HE encryption schemes, or SMPC cryptographic primitives)</li> <li>• Explores networking type and NN architecture in SMPC to better evaluate the testing environments and results</li> <li>• Reviews HE settings and protocol design trends.</li> <li>• Perform experiments using OSI of the selected approaches.</li> </ul> |
| PPTs                             | HE, SMPC, DP when it intersects with HE / SMPC   | Only HE and SMPC  | Only HE and SMPC  |
| Categorization of surveyed works | No categorization for works in privacy-preserving training   | Categorizing works based on cryptographic primitives and techniques for both HE and SMPC  | <ul style="list-style-type: none"> <li>• HE: Categorizing works based on HE techniques</li> <li>• SPMC: Categorizing works based on the number of parties involved in the privacy-preserving training protocol two, three, four-party computation</li> </ul>  |
| PPML works                       | 2 for HE and 5 for SMPC  | Study 12 works that support private training  | 9 for HE and 17 for SMPC  |
| Reproducing results              | No   | Focus on private inference and 4 private training results reported  | Focus on private training phase   |
| Pros                             | Cover both training and inference phase  | <ul style="list-style-type: none"> <li>• Cover both training and inference phase</li> <li>• Evaluation of many works on the MNIST and CIFAR dataset in the private inference phase</li> <li>• Identifying various challenges and open problems in current cryptographic NN computation</li> </ul> | <ul style="list-style-type: none"> <li>• Cover the private training phase using HE and SMPC</li> <li>• Contain the evaluation of the datasets and OSI of the surveyed works in the private training phase</li> <li>• Identifying various challenges and reproducibility problems regarding secure NN training</li> </ul>  |
| Cons                             | <ul style="list-style-type: none"> <li>• Quite brief in survey of works in PPML training phase</li> <li>• No categorizations for PPML training works</li> <li>• No evaluation of datasets or implementations</li> </ul>  | Only briefly study the private-training phase   | Does not cover the inference phase  |

in real-world applications, as modern solutions require relatively large datasets for proper training and various layers for different task applications. Aside from theoretical aspects, we also analyze various experimental results, which are important for the selected method, such as the training time and accuracy for both methods, the communication overhead for SMPC approaches, and the testing environment specifications and complexity for HE approaches.

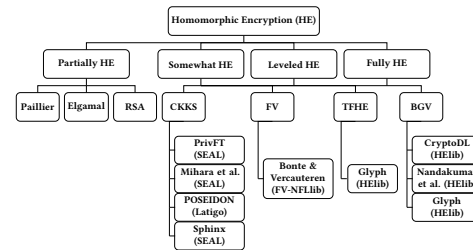
To analyse the aforementioned aspects in SMPC and HE PPML implementations, we aimed to identify literature encapsulating them in experiments or methodology. The PPML implementations were identified by analysing works published in ML or information security-focused venues and by cross-checking the referenced works in those papers. Our main focus was to identify and categorise literature using different libraries trained and tested on various datasets, and implementing varied techniques, schemes and settings. As such, we identified 9 HE-based PPML approaches covering the various schemes and settings in the HE domain and 17 SMPC-based PPML implementations. To the best of our knowledge, these 26 works show the primary areas of focus in cryptographic PPML, by encapsulating the different aspects covered previously.

### 3 STATE-OF-THE-ART APPROACHES

#### 3.1 Secure Training using HE

As covered in the preliminaries, implementation of HE in PPML models depends on mathematical operations that the HE scheme can use and on whether the HE technique works on binary [41], integer [55] or approximate numbers [40] (see Table 2<sup>3</sup>). The overall performance and cost of the Neural Network (NN) will vary depending on the HE scheme selected. There is no single HE approach that outperforms all others; rather, performance varies from application to application depending on the HE scheme selected [145]. To further clarify the implementations covered, we have categorized them according to the underlined HE scheme (see Figure 2).

**FV-based implementations:** Bonte and Vercauteren [30] – is a Logistic Regression (LoR) [80] model based on the SHE implementation of the FV scheme. The implementation uses the fixed Hessian method at a low depth to construct an HE algorithm capable of privacy-preserving logistic training. To accomplish this, the researchers show that a practical algorithm can be constructed by using the Simplified Fixed Hessian (SFH) method. When compared to Matlab’s `glmfit` function [9], the SFH algorithm produces equivalent accuracy, while providing superior security for the data. Alongside FV schemes there is BGV, another HE scheme allowing



**Figure 2: HE Taxonomy**

performing homomorphic operations on integers arithmetic. The core difference between them is their plaintext encoding. Namely, BGV encodes the messages starting from the least significant bit, while FV starts from the most significant bit.

**BGV-based implementations:** CryptoDL [70] – is an SHE implementation that uses the BGV scheme to encrypt data. The open-source library HElib [8] is used for implementing the BGV scheme and also contains various optimizations for HE. CryptoDL is one of the first implementations, providing a solution for NN training using encrypted training data through HE. In their work, the researchers show how to locate approximations of low-degree polynomials and how to approximate non-linear activation functions, such as ReLU, sigmoid, and tanh. The proposed implementation was trained and tested on various datasets, such as CIFAR [85],

<sup>3</sup>Due to space constraints, all the tables are moved to the appendix section.

MNIST [88], and UCI [22]. The research is compared to CryptoNets [62] (an HE implementation of NN classification) and SecureML [39]. CryptoDL [70] outperforms the other implementations both in terms of accuracy, classification throughput, and communication overhead. However, it only states a training time of 10476.29 seconds for MNIST implementation and doesn't provide further comparisons with other implementations.

Nandakumar *et al.* [107] – is an FHE implementation of PPML training. The model is trained in a non-interactive way and utilizes the BGV scheme through HELib. The research improves on prior implementations by optimizing the way ciphertext is packed to reduce bootstrapping and encourage parallel training. The implementation uses the MNIST dataset for training and classification and protects the client privacy by encrypting both the training data and the model parameters. According to the research results, the implementation reaches 96.4% and 97.8% accuracy on different NN architectures, when training for 50 epochs on the plaintext, while the computation time for a single mini-batch of 60 training samples varies between 40 minutes and 9.4 hours with their optimized packing and parallelization.

Research on the BGV scheme led to Cheon *et al.* to propose a major improvement to the scheme expanding the homomorphic operation to floating point arithmetic in the CKKS scheme [40]. This scheme has greatly impacted the development of new PPML works and is used by most SoTA works.

**CKKS-based implementations:** PrivFT [19] – is an LHE implementation based on the CKKS LHE scheme. It allows the use of a GPU for faster training and faster inference. The GPU implementation is achieved by using the residual number system (variant of the CKKS scheme) and by using CUDA 10 [1]. The implementation was tested on various datasets, such as Yelp Dataset [73], AGNews news topic classification [66], IMDB movie reviews [3, 98] and DBpedia ontology classification [45], to name but a few. PrivFT compared the accuracy of the model with XLNet [153], BERT ITPT [138] and ULMFit [72] on 4 datasets (Yelp, AGNews, IMDB and DBpedia) and noted that the accuracy of PrivFT was lower than the other models. The largest accuracy difference noted was on the IMDB dataset, when compared to the XLNet model (91.49% versus 96.21%). However, the main advantage of PrivFT is its faster training, boosted by GPU usage, which accelerates training by approximately 2.2x.

Sphinx [142] – is a CKKS LHE scheme used for online training and inference on the cloud. The code is implemented with the Microsoft SEAL library for homomorphic operations and makes use of KANN<sup>4</sup> to implement ML models in C. Sphinx combines various improvements, such as batch packing, made in HE PPML training and inference to greatly reduce the communication overhead and computational complexity of ML operations. The authors improve HE multiplications by reducing the amount of rescaling and relinearization operations. They also use forward propagation caching and introduce a different encryption method, called zero encryption, to reduce communication costs. The implementation is tested on MNIST and CIFAR-10 datasets and shows reduced communication costs and computational complexity when compared to baseline HE. It reports higher throughput and lower latency than works such as SecureNN [146], SecureML [103] and CryptoDL [70].

<sup>4</sup><https://github.com/attractivechaos/kann>

Mihara *et al.* [100] – is a CKKS-based LHE implementation using the Microsoft SEAL library. The main optimization of this implementation is that it provides a novel weight matrix packing method, increasing the speed of the training phase without losing accuracy during the inference phase. The packing method in question packs weights in a matrix diagonally instead of in a row, which keeps the amount of operations low as the packing does not require multiplication. This in turn reduces the complexity of the circuit. The authors note that their packing method reduces the time of training for one iteration from 28.47s (row packing) to 9.25s (diagonal packing). They test and compare their implementation with the same architecture plaintext NN and receive similar results (98.05% accuracy) to their ciphertext model (98.47%).

POSEIDON [130] – is a hybrid PPML implementation that makes use of federated learning and LHE techniques to produce Multiparty Homomorphic Encryption (MHE). POSEIDON's MHE implementation uses the CKKS scheme and is an extended version of the Mouchet *et al.* [104] implementation. The implementation provides confidentiality for: (i) Training data, (ii) Model details and (iii) Evaluation data. POSEIDON's accuracy and training time is similar to other SotA SMPC techniques, such as SecureML [103], SecureNN [146] and Falcon [147]. When compared to other HE techniques, like Nandakumar *et al.* [107] and CryptoDL [70], it outperforms them on a relatively large margin.

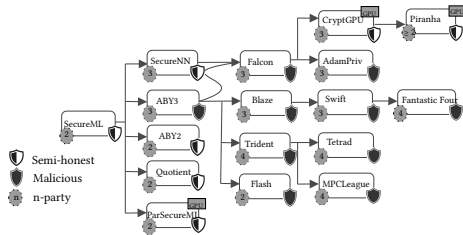
The core limitation of FV, BGV and CKKS works is that these schemes have very expensive and impractical bootstrapping operations resulting to increased computational complexity. To address this Chillotti *et al.* [41] proposed TFHE, a scheme that speeds up bootstrapping operations.

**TFHE-based implementations:** Glyph [95] – is a hybrid FHE approach, using two different cryptosystems, namely TFHE and BGV. TFHE is used to implement non-linear activation functions and BGV scheme is used to perform multiply-accumulate operations. Another benefit, is that switching between the two cryptosystems results in significantly reduced training times compared to other PPML techniques. In the results, the researchers showed that the accuracy of their implementation was comparable to Chimera [31] – another hybrid HE technique. Chimera is similar to Glyph as it switches between the TFHE and BFV. An additional benefit of Glyph is that the training time (8 days) for the MNIST dataset is 2.6× faster than the Chimera implementation (28.6 days). Furthermore, Glyph was slightly more accurate than Chimera (98.6% vs 97.8%). The researchers also compared Glyph with the Nandakumar *et al.* [107] implementation and showed that Glyph was significantly faster during training (8 days compared to 13.4 years).

### 3.2 Secure Training using SMPC

Over the past years, various methods focused on constructing SMPC protocols with different properties and settings. However, as listing all the relevant techniques exceeds the scope of this work we recommend a well-written and friendly introduction to SMPC [54]. Nevertheless, we intend to review the most important SMPC approaches that perform PPML training among multiple parties and show their relationships in Figure 3. These works are discussed in detailed below. We categorize SMPC techniques based on the number of parties involved in the protocol. This categorization can

vary from a 2-party protocol, where training is conducted on two non-colluding parties, to a 4-party protocol, where the protocol can be carried out by four parties. The main motivation for increasing the number of parties in the SMPC protocols can be twofold: (i) aiming for more efficient computations and collaborative tasks with the same amount of corruptions as in lower party variants, or (ii) improving security through increased resilience against more potential malicious parties. However, it is important to note that the specific security guarantees may vary depending on the chosen protocol and configuration.



**Figure 3: SMPC Taxonomy – illustrates comparative framework employed in our study, showcasing performance evaluations among protocols. For example, Falco against SecureNN and ABY3, and SecureNN against SecureML.**

**Two-party Computation:** The 2-party computation is a representative of SMPC [63]. Many 2-party SMPC based ML models have been developed among which SecureML [103], proposed by Mohassel *et al.*, was the first privacy-preserving protocol for efficient NN training. It is based on SS protocol where the data owner distributes private data among two servers. Compared to previous SotA frameworks in PPML, SecureML is much faster than the protocol implemented in [57, 110], however, it is an order of magnitude slower than the privacy free counterparts. Though SecureML excels in certain aspects of PPML, its limitations in WAN settings (high number of interactions and communication overhead) prompt the exploration of alternative frameworks. QUOTIENT [17] emerges as a compelling choice, showcasing superior speed and accuracy in both WAN and LAN settings. Its core idea lies in ternarizing the network weights into integer values of  $\{-1, 0, 1\}$  upon which the SMPC-aware NN training protocols are developed. Compared to SecureML, training with QUOTIENT is 50× faster in both WAN and LAN setting. By implementing ML features like adaptive gradients and the normalization approach, QUOTIENT reaches the level of accuracy of SecureML relatively fast – in less than two epochs. Furthermore, for the same network in SecureML, QUOTIENT achieves an overall accuracy of 99.38%, a 6% improvement over the SecureML (93.4%). In order to make a 2-party SMPC technique even more efficient, a different approach is taken by ParSecureML [39]. ParSecureML, leveraging GPU-based parallelization, offers a substantial speedup, presenting a noteworthy alternative for efficient PPML. It consists of three major parts: (i) Profiling-guided adaptive GPU utilization, (ii) double pipeline design for intra-node data transmission between CPU-GPU and (iii) compressed transmission for inter-node communication between two servers. These three components are integrated in order to enable 2PC on GPUs. Compared to SecureML [103], the authors show that ParSecureML achieves an

average of 32.2× speedup. Another notable work in the field of 2-party PPML is ABY2.0 [117]. ABY2.0 proposes a mixed-world SMPC protocol between Arithmetic, Boolean and Yao sharing, based on which a NN training protocol was built. For NN training, ABY2.0 has 2.7-3.46× and 2.4-2.8× online throughput improvements for LAN and WAN settings respectively.

In a 2-party SMPC protocol, where only two parties are involved, the threat model considered is a weaker semi-honest threat model (see Table 3). This model it assumes parties will follow the protocol and this might not always be the case in the real world. If one party is malicious, the system may not work effectively. To address this limitation and take full advantage of the benefits of SMPC, aimed at facilitating secure collaboration among multiple distrustful parties, researchers are exploring ways to involve more than two parties. The primary goal of adding more parties is twofold: firstly, researchers aim to improve the training process by incorporating a larger dataset – more parties participating implies more data for training. Secondly, this expansion is intended to address a more realistic malicious threat model. This widens the applicability and enhances the security of SMPC in real-world scenarios.

**Three-party Computation:** In 3-party SMPC, only one protocol, CryptGPU [140], exclusively considers the weaker semi-honest threat model. Others take into account both threat models. For instance:  $ABY^3$  [101] designs techniques for encrypted training of DNNs in the 3-party settings with a single corrupted server. This is a mixed protocol framework for ML, which uses SMPC techniques and offers efficient support for fixed point arithmetic computation, improved matrix multiplication (to reduce the amount of communication) and an efficient piece-wise polynomial evaluation technique.  $ABY^3$  experiments on both inference and training for LR, LoR and NN models. For training NNs,  $ABY^3$  is 55,000× faster than the 2PC solution of SecureML. Another significant contribution in the 3-party SMPC settings is *SecureNN* [146], designed to ensure privacy against one malicious corruption and privacy and correctness against one semi-honest corruption. Prior works to SecureNN use boolean computation and Yao’s Garbled Circuits (GC) to construct functions, such as ReLU, Maxpool and their derivatives. This results in increased communication cost. SecureNN’s proposed protocols for Boolean computation require much lesser communication overhead than the cost of converting to a Yao encoding and executing a GC. Thanks to protocol efficiency, SecureNN is the first work to privately train a Convolutional Neural Network (CNN) [20] with an accuracy of higher than 99% on MNIST. Compared to SecureML, SecureNN is 79× faster in the LAN (latency 0.22 ms, bandwidth 625 MB/s) setting, and 553× faster in the WAN setting (latency 58 ms, bandwidth 40 MB/s). In the 3-party LAN setting, SecureNN outperforms SecureML in training time by 7×. Moving forward, Patra and Suresh proposes a PPML framework named *BLAZE* [118] in the 3-party setting, tolerating one malicious corruption over a ring ( $Z_{2^r}$ ). It consists of a data-independent preprocessing phase, used to perform a relatively expensive computation, and a fast input-dependent online phase. The authors benchmark the performance of the framework over  $ABY^3$  for training LR and LoR models. Over a dataset with a feature size of 784 and batch size of 128, training in Blaze gains about 4× more throughput in the preprocessing phase for both LR and LoR. For the online phase, Blaze gains 145.35× and 31.89× throughput for LR and LoR respectively compared to

*ABY*<sup>3</sup>. Shifting focus to *Falcon* [147], which is an end-to-end 3-party protocol for efficient private training and the inference of large ML models. By combining techniques from SecureNN and *ABY*<sup>3</sup>, Falcon constructs improved protocols for various private ML operations, such as convolutions, matrix multiplications, private comparison, ReLU, and its derivative, division, and batch normalization. Compared to other private training frameworks, Falcon is about 4.4× faster than *ABY*<sup>3</sup> and 6× faster than SecureNN. Furthermore, Falcon achieves 2× to 60× less communication overhead than *ABY*<sup>3</sup> and SecureNN. Similarly, *SWIFT* [83] relies on an efficient, malicious-secure 3PC framework, that works over rings ( $\mathbb{Z}_{2^t}$  and  $\mathbb{Z}_{2^1}$ ). SWIFT provides Guaranteed Output Delivery (GOD) in the honest majority setting. The protocols work in the preprocessing (offline-online) model. One highlight contribution is the dot product protocol that achieves communication cost independent of the input vector sizes. The authors benchmark their method using LoR (for training and inference), LeNet [89] and VGG16 [136] NNs (for inference). Authors compare LoR model’s training in a 3-party setup with BLAZE, finding similar performance and improved security.

Adam in Private [26] addresses specific tasks within a 3-party setting and specializes in tasks such as integer division, exponentiation, inversion, and square root extraction within the context of DNNs. To demonstrate the proposed protocol’s scalability, the authors perform measurements on DNNs architectures, such as a 3-layer fully-connected network introduced in SecureML, AlexNet and VGG16 on two datasets –MNIST and CIFAR10. The experiments are carried out in both LAN and WAN and the results are compared to Falcon. The implementation of the Adam optimization algorithm allows the framework to converge faster and requires fewer epochs compared to Falcon. In terms of training time, Adam in Private is 3.2× to 6.7× faster than Falcon for the 3-layer NN, about 12× to 14× faster for AlexNet and 46× to 48× faster for VGG16.

While the above works demonstrate a CPU-only implementation, there are a few works that explore GPU assisted computation within the standard 3-party setting such as *CryptGPU* [140] and *Piranha* [149]. *CryptGPU* operates, where all inputs are secretly shared among three non-colluding servers executing the protocol. It is built on top of PyTorch [116] and Crypten [81]. Experiments show, that the GPU-based protocol can have a 150× speed up over the CPU-based protocol for a convolution operation and up to 10× the speed up for non-linear operations. The increased efficiency of GPU implementations, enables the framework to privately train big networks such as LeNet, AlexNet [86], VGG16 over MNIST, CIFAR-10 and ImageNet datasets. Compared to Falcon, *CryptGPU* achieves 6× to 36× improvement for private training. Though showing great progress in terms of GPU adoption, *CryptGPU* is tailored for a specific SMPC protocol and can only demonstrate full end-to-end training on simple networks such as AlexNet. To improve upon this, *Piranha* [149] proposed a more general GPU-based framework that can be used to implement other SMPC protocols. *Piranha* consists of three layers: On the device layer, *Piranha* implements a data abstraction that manages vectorized GPU data and integer GPU kernels to support acceleration for integer-based computation needed for SMPC protocols. *Piranha*’s protocol layer uses the device layer to implement functionality for different SMPC protocols, namely SecureML (2-party) [103], Falcon (3-party) [147],

and FantasticFour (4-party) [47]. In the application protocol, *Piranha* provides typical privacy-preserving layers for NNs such as the linear and convolution layers, pooling operations, the ReLU activation function, and layer normalization. The SMPC protocols implemented with *Piranha* exhibit 16-48× training time speed up compared to their native CPU implementations. *Piranha* is also the first PPML framework capable of training a big NN such as VGG16 (with over 100 million parameters) over the CIFAR10 dataset in a short amount of time (less than a day and a half).

**Four-party Computation:** As we move forward, the ongoing evolution of 3-party SMPC protocols not only enhances efficiency, security, and applicability in collaborative computations but also lays the foundation for exploring more complex scenarios. This progress is evident in recent developments within 4-party SMPC protocols, where innovative solutions address diverse challenges in secure computation. For example: *Trident* [38] proposes a framework in the setting of 4-parties with at most one corrupt participant over the ring  $\mathbb{Z}_{2^t}$ . The sharing protocols follow the offline-online phase paradigm and are used to construct a mixed-world framework between the binary, arithmetic and GC worlds. Compared to *ABY*<sup>3</sup> [101] (for LR), training with *Trident* is 4.88× to 251.84× faster in LAN and 2× to 2.83× in WAN. For the LoR model, *Trident*’s improvements range from 5.95× to 67.88× in LAN and 2.71× to 2.96× in WAN. Training the NNs, *Trident* is from 3.56× to 62× faster in LAN, and 2.97× to 3.56× faster in WAN. Building on the foundation laid by *Trident*, *MPCLeague* [139] operates within a 4PC setting over the ring  $\mathbb{Z}_{2^t}$ , ensuring support for an honest majority with at most one corrupted party. The framework combines arithmetic, boolean and garbled worlds with efficient end-to-end conversions between them. The efficiency of *MPCLeague* in private NN training is benchmarked using networks such as LeNet and VGG16. Compared to *Trident* [38] based on runtime, communication, and cost, *MPCLeague* offers better performance overall.

In the same year, Koti *et al.* introduced *Tetrad* [84], a 4-party setting designed to tolerate at most one active corruption over the ring  $\mathbb{Z}_{2^t}$ . *Tetrad* proposes a mixed SMPC protocol that supports robustness and fairness. Similar to *MPCLeague*, the authors of *Tetrad* compared their results with *Trident* considering training time, communication and monetary cost, as this captures the effect of the total runtime and communication of the parties. Experimental results show that *Tetrad* is 3-4× faster than *Trident* and achieves approximately 30% better results in terms of monetary cost. To obtain security against malicious parties, Dalskov *et al.* proposed *Fantastic Four* [47], an actively secure 4-party protocol for corruption over a ring  $\mathbb{Z}_{2^k}$ . The protocol tolerates one active corruption and satisfies security with abort, however, it also provides GOD with some extensions. The protocol guarantees to identify a semi-corrupt pair and remove one party in the pair, then proceeds to an actively secure 3-party protocol with abort. If the 3PC protocol succeeds, then the output is produced, and the computation is finished. If the 3PC protocol also aborts, then the protocol knows that it removed an honest party and will abort the remaining party in the corrupt pair. It then proceeds to a passively secure 2-party protocol. The protocol is applied for training fully-connected NNs with 1-3 dense layers and a LoR model on the MNIST dataset. For the LoR model, *Fantastic Four* achieves 172.05× faster training time than *SWIFT* in 4PC setting, and 84.24× in 3PC setting.

To summarize, the choice among 2-party, 3-party and 4-party SMPC depends on the specific requirements and characteristics of the collaboration at hand. The lower party like the 2-party SMPC may be favored for simplicity and lower computational overhead, whereas the more intricate 4-party SMPC offers advantages such as enhanced collaboration, security, and flexibility in situations involving more than two parties. Adding more parties may offer better security and efficiency, though it necessitates more servers, more communication overhead, more efforts to coordinate and increased complexities in the algorithms.

## 4 EVALUATION

### Reproducible Research: An Unfulfilled Dream

In this section, we evaluated both the performance and security of several PPML works. At first, we intended to benchmark and compare 23 impactful PPML techniques based on HE and SMPC. As a result, we made an effort to reproduce the results for all these works. However, this proved challenging because most of the implementations did *not* provide open-source code. We hence tried to reimplement some of the schemes following the instructions given in the respective papers.

However, implementation of HE and SMPC techniques proved to be monumental tasks and we were unable to reproduce accurate results, because of difficulties, such as time constraints, hardware limitations, and library issues. Although a few works have OSIs, they lack proper documentation, making it challenging to use those works. Also, comparing works that are exclusively designed for either CPU or GPU can be challenging because it restricts cross-platform evaluation. Additionally, given our focus on SMPC-based techniques, variations in the number of parties involved, such as two-party or three-party computation, can result in different performance characteristics, making comparisons more complex. While the ideal scenario would involve a universal framework evaluation under various settings, this would demand substantial engineering effort and computational resources, which are currently beyond our capacity. Consequently, we were only able to reproduce the results of 4 PPML techniques, the details of which are given in Table 5. Despite the fact that solutions are comparable in terms of metrics -i.e. computational and communication costs accuracy and memory usage- unfortunately, not all solutions provide said metrics (see Table 5), thus rendering direct comparisons difficult. Despite the mentioned difficulties, we hope to give some valuable insights to researchers who wish to further expand the area by covering the results of SotA in the field and mentioning some of their existing limitations.

### 4.1 Experiments

**HE Protocols:** In this section, we overview results from available HE frameworks. We also cover aspects relating to their performance and their availability, while equally focusing on the specifications used by the researchers during testing.

*Testing environments.* As covered in the preliminaries section, HE is computationally expensive to properly implement and test. This fact can be corroborated by the implementations covered as the majority used high-performance server processors, such as the Intel Xeon series and considerable amounts of RAM (i.e. ranging from 48 to 256GB). An exception to the high requirements is the implementation made by Mihara *et al.* [100] requiring only consumer-level hardware, such as the Intel Core i7 and up to 32GB of RAM. All of the implementations run a Unix-based operating system, such as Linux, through Ubuntu or ArchLinux. As can be seen in Table 4, the high computational requirements of HE create an entry barrier making it impossible for most users to use HE technology outside commercial companies and academia. However, HE can still be used in cloud services. In this case, most users can use the services, while keeping their personal information private.

### Reproducible Research: Myth Buster #1

When overviewing the availability of HE frameworks for result replicability and further testing, it can be noted that most frameworks do *not* release an OSI of their code. Hence, the majority of frameworks extensively cover their implementation (a) by providing libraries used for the implementation (see Figure 2 and subsection 4.2) and (b) by defining the functions and model parameters for others to see. Most works provide a pseudocode and algorithms for their implementation, however, none of the surveyed HE works provided openly available code. As mentioned before, it is possible to recreate the core aspects of private implementations with the tools provided in the papers. However, the main drawback in rewriting the code is that the resulting code might produce large deviations from the original results, due to the naturally occurring differences in algorithms. Another aspect to be taken into account is the amount of time needed to recreate the code and implement HE into ML techniques. *As such the lack of OSIs creates possible gaps in reaffirming results and advancing science.*

*Performance.* An important topic when covering ML techniques is overall performance both in terms of accuracy and training time. Hence, it is important to overview the performance of HE techniques introduced into ML and compare their results. Following the results listed in Table 4, one can notice that the accuracy of almost all the frameworks is more than 90%. There are two exceptions regarding accuracy: (a) the hybrid technique POSEIDON [130] and (b) the implementation proposed by Bonte and Vercauteren [30]. For POSEIDON, accuracy is slightly below 90% on the MNIST dataset (*Accuracy: 89.9%*), however POSEIDON has been tested on multiple datasets, including the Breast Cancer Wisconsin (BCW) (*Accuracy: 96.9%*), Epileptic Seizure Recognition (ESR) (*Accuracy: 90.4%*) and default of credit card clients (CREDIT) datasets (*Accuracy: 80.2%*) [130]. Bonte and Vercauteren [30] tested their implementation on two datasets: (i) the iDASH genomic dataset [141], which showed lower accuracy when trained with different parameters (*Average accuracy: 64.37%*) and (ii) a private financial dataset, whose accuracy was higher on average when trained with different parameters (*Average accuracy: 92.38%*). Other implementations,

such as CryptoDL [70] show high levels of accuracy – reaching up to 99%. This, in turn, shows that HE techniques can provide the same level of accuracy as their plaintext counterparts, when compared for the same dataset and NN architecture.

However, the main concern about HE is the high computational complexity as mentioned in prior sections. That is why when over-viewing the training times of all of the frameworks we can notice, most implementations can take hours or even days to fully train on different-sized datasets. When comparing HE techniques, which used the MNIST dataset for training, we observe that Nandakumar *et al.* [107] takes the longest to train (approximately 60K-84K hours) followed by Sphinx [142] (approximately 1560 hours). The extremely long training time for Nandakumar *et al.* is theoretically calculated by combining the reported training time in hours for a single mini-batch (reported to vary between 0.667h - 9.4h), multiply by the total amount of mini-batches (reported to be 1,800) and then multiplying by the amount of epochs required to train a plaintext model of the same size (reported to be 50) [107]. Similarly, training time for Sphinx [142] is calculated by taking the training time for a single batch (reported to be 0.108h) multiply it by the amount of batches (calculated by dividing the total amount of training samples by the batch size  $\frac{60000}{500} = 120$ ) and then multiply by the amount of epochs (reported to be 120 for MNIST). However, it is important to note that the model used in Nandakumar *et al.* [107] contains only 3 fully connected layers with a sigmoid activation function, while Sphinx contains 2 convolution, 2 average pooling and 2 fully connected layers with ReLU activation functions. Nandakumar *et al.* set a solid baseline for further research into FHE applicability in PPML environments. Since this was one of the first attempts to train an NN in a privacy-preserving way using HE, the training time is the longest of the covered implementations. The performance of newer implementations, such as POSEIDON and Glyph, increased together with the knowledge on PPML techniques. Training times in these implementations are more manageable. The shortest training time is reported in POSEIDON [130] (approximately 1.47h) and involves using FL and HE. However, even though the training time of HE techniques has dropped considerably in the past years, they are multiple times slower to train, when compared to plaintext counterparts. This increase in training time may be leading issue impeding consumers and suppliers moving towards PPML, even though HE techniques provide SotA privacy for their data.

#### Reproducible Research: Myth Buster #2

Our initial goal was to replicate the results for all SMPC works that we surveyed; however, we were able to replicate the results of a few works for local hosts and did not conduct any experiments for LAN or WAN. The reason for this is that we ran into a number of problems, while attempting to reproduce the results for these works such as lack of availability; only 8 of the 15 works provide OSI. Also, some works, that provide OSI, lack proper documentation for implementing the code.

**SMPC Protocols:** We evaluate SMPC frameworks with an open-sourced code on the MNIST dataset over popular network architectures in PPML domain. The results are reported in Table 5.

**Testing Environment.** To reproduce the results for SMPC, first we created four Virtual Machines (VM) on a locally deployed cloud. The experiments are conducted using four VMs equipped with Ubuntu 18.04.2: LTS, Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz, 16 CPUs, 64 GB RAM and 256 GB Disk. As our VMs are not equipped with dedicated graphic cards, we cannot experiment with works such as ParSecureML and CryptGPU.

**Performance.** Below we will discuss the results as well as the issues we faced while implementing the surveyed papers.

**SecureML:** Although SecureML did not provide an OSI of their work, we did locate one GitHub repository [135] referring to the implementation for SecureML. First and foremost, the author made a commendable contribution by implementing SecureML and provide it as open-source is commendable; however, building SecureML from this repository is not simple. Second, this repository only contains code for secure two-party LR; there is no implementation for LoR and NNs, while in the paper, the author proposes a secure two-party approach to train LR, LoR and NNs. Finally, only one performance metric (accuracy) is considered, while other performance metrics, such as communication cost, and time are ignored. For this survey, we reproduce the SecureML results only for the local host, receiving an accuracy of 92.98% for 100 epochs.

**SecureNN:** The authors provide an OSI of SecureNN that includes ML training and inference. We only performed experiments for training where three parties locally trained the LeNet model. The results reproduced only consist of training time and communication cost (see Table 5). The training time for LeNet (1 epoch) is 136.533sec, and the total communication cost is 21058.3MB, which is quite efficient in terms of time and cost.

**Falcon:** Falcon produces fast training, however, the code provided does not produce accuracy information after training and is limited to training time and communication. The communication overhead for LeNet may also be erroneous, as running the code shows that first-party communication is 3346.56MB, hence bigger than the total communication attributed to the whole protocol as shown in Table 5 (only 1800 Mb). Based on this information, the communication cost for LeNet can be approximated to around 10,000MB for 3 parties. We submitted this issue, however, the repository seems to no longer be under active maintenance.

#### Reproducible Research: Myth Buster #3

*Quotient, Blaze, SWIFT, Adam in Private, Trident, MPCLeague, Tetrad*, do not provide OSIs, and others (*Fantastic Four, ABY*<sup>3</sup>) are extremely difficult to implement. Also, the authors of *ParSecureML and CryptGPU* ran their experiments on high-end accelerators such as NVIDIA Tensor Cores and NVIDIA Tesla V100 GPUs. As our VMs are limited to only CPUs, we were unable to run and reproduce the results of these papers.

**AriaNN:** We reproduced the results for AriaNN (Network 1 and 2), where the accuracy was 97.96% and 97.92%, respectively as in Table 5 compared to 98.0% and 98.3% from [127]. However, for LeNet, we only got 89.36% compared to 99.2% reported in [127]. We also got longer training times, most likely due to our less capable machines.

On the other hand, we calculated and reported the communication for private training of each epoch, which was not reported in [127].

## 4.2 Analysis of HE protocols

We analyze and summarize various aspects of HE protocols introduced in subsection 3.1.

**Security:** One of the most important factors to take into account, when analyzing and assessing various secure computation protocols. In this context, almost all of the HE protocols show security strength of approximately more than 80 bits. This is the minimum required strength for non-federal government information, but does not provide enough security as documented in the NIST Special Publication 800-57 [28]. As mentioned in that report, the security strength defines “a number associated with the amount of work (i.e., the number of operations) that is required to break a cryptographic algorithm or system”. When looking over the works analyzed, the protocol proposed by Bonte and Vercauteran [30] estimates, that the security strength is equal to 78 bits, namely below the previously mentioned minimum threshold. Other protocols such as the one proposed by Nandakumar *et al.* [107] and Glyph [95] note a security strength of 80 bits, while most others [70, 130, 142] approximate a security strength, that is comparable to AES-128. The largest documented security strength is produced from the PrivFT [19] protocol, which estimates a strength of 140 bits.

Unlike the other HE implementations, Sphinx [142] makes use of DP to increase its resilience to data reconstruction attacks. The authors test their implementations resilience to a gradient matching attack, which aims to recover input images and their labels from the intermediate gradients [142]. From their testing, they show that Sphinx outperforms an equivalent DP-only defence mechanism irrespective to the chosen privacy parameters.

Since HE is a relatively new way of implementing PPML, its security against ML attacks, such as poisoning attacks, model inversion attacks or reconstruction attacks, has not yet been tested or documented to the best of our knowledge (aside from the gradient matching attack covered in Sphinx [142]). However, as mentioned by researchers, one of the main issues for HE in PPML is finding a suitable compromise between efficiency, accuracy and privacy provided by HE algorithms [152], that would enable the HE-based protocol to provide effective privacy guarantees without compromising the computation times or accuracy of the ML algorithm.

**Encryption Schemes and HE libraries:** Different HE schemes allow for tailored use which increases the accuracy and efficiency of the selected use cases. Because of this each analysed paper made use of the common HE schemes discussed in Appendix A. BGV is used by Nandakumar *et al.* [107] and CryptoDL [70], while Bonte and Vercauteran [30] use the FV scheme. PrivFT [19], Sphinx [142], Mihara *et al.* [100] and POSEIDON [130] use CKKS, while Glyph [95] uses the TFHE encryption scheme. Each paper made use of HE libraries to implement operations on encrypted data and developed their solutions through the use of them.

In terms of HE libraries, the most commonly used is Microsoft SEAL [7] It supports BFV and CKKS schemes. As can be seen in Figure 2, SEAL is used in [19, 100]. HELib [8] another popular and early FHE library represented in this work, supports the BGV and CKKS schemes. HELib is used in paper [70, 95, 107]. Both HELib and

SEAL are widely used for binary plaintext spaces. They construct binary circuits to compute the desired functions over encrypted data. They do, however, provide the option of a larger plaintext space in situations, where the functions can be evaluated more efficiently, when represented by a modular arithmetic circuit [18]. The FV-NFLib [6] library only supports BFV scheme. According to [82], FV-NFLib is faster than SEAL library. However, FV-NFLib does not support high-level circuits (only up to 6 levels). Therefore, the use of FV-NFLib is recommended for small circuits, while SEAL for larger ones. Lattigo [10] also supports BFV and CKKS schemes and their respective multiparty version. It is written in Go language and performs similarly to cutting-edge C++ libraries (HELlib, FV-NFLib, SEAL). Other recent and promising FHE libraries include OpenFHE [11] and the Zama.ai libraries [12]. OpenFHE is a C++ library and supports most modern FHE schemes, such as BGV, BFV, CKKS, FHEW, and TFHE, as well as multiparty extensions for BGV, BFV and CKKS. One of the libraries’ major focuses is usability. This is achieved by streamlining the parameter selection process and using the same common API for functions in different schemes. Zama.ai hosts the libraries’ TFHE-rs [13] and Concrete ML [14]. Both of the Zama libraries focus on allowing non-experts of cryptography to implement FHE into their applications.

**Computational Complexity:** The biggest challenge for HE is the high computational complexity required to train NNs. This has held HE back from being included in modern-day use. With current-day improvements to all HE algorithms and hardware development, the required computations and the time it takes to compute them have vastly improved compared to older HE implementations. However, despite said improvements the computation time is still comparatively high and requires high-end machines to implement HE algorithms. This is noted in the results provided by researchers, who propose HE implementations: the training time expands from multiple hours [70, 77, 107, 109, 130] to multiple days [19, 95] for comparatively simple datasets, such as MNIST requiring little-to-none preprocessing. Current-day plaintext implementations of NNs using MNIST can train hundreds of epochs on complex NNs in the span of a couple of minutes [58], while most HE solutions require notably more time for  $\leq 10$  epochs. As noted by researchers, the training time vastly increases, when training on more complex datasets, such as CIFAR-10 or CIFAR-100 [70, 130].

**Adversarial Model:** The adversarial model for HE-based PPML works were defined as having a semi-honest adversary. In this case, the adversary can only passively listen and gather information from an available source, such as the dataset. This model is plausible for various real-world applications, where a client would want to store and classify data in a Cloud Service Providers (CSP). However, new discoveries could be envisaged regarding security of HE protocols, if assumption is changed to malicious adversaries, which have more tools for breaking security of the algorithms.

**Scheme Applicability:** Given the computational complexity of HE, it is important to identify use cases where HE can effectively operate within its limitations. As such, HE provides the needed capabilities for directly implementing privacy-preserving MLaaS. HE requires high computational capabilities, which can be provided by a CSP. A user would be able to design and train their own model, despite their computational capabilities, by hosting a server on a CSP and providing the needed training HE data. The main decision for the

user is choosing the correct HE scheme for each task. Both BGV and FV are malleable to a variety of applications, but require quantization from floating point to integers. This can cause loss of precision and reduced accuracy. As a result, these schemes are best suited for use cases where precise values are unnecessary (i.e. image classification). On the other hand, CKKS allows operations to be accurately performed on floating point values and can be more suited for implementing non-linear activation functions through polynomial approximation. This allows CKKS to be used in precise tasks involving one-dimensional medical data as well as in object detection. Lastly, TFHE benefits from having fast bootstrapping operation and allows simplified implementations of non-linear functions as they would not require polynomial approximation. Through TFHE users would be able to construct more complex models, than the other schemes, but can face issues of scalability because of look-up table storage costs. This leads TFHE to be more suited for natural language processing tasks and complex data structures.

### 4.3 Analysis of SMPC protocols

While considerable progress has been made regarding the efficiency of SMPC protocols, some of the current approaches remain computationally expensive and do not scale well with the types of NNs typically used in modern ML systems. Another significant problem is the requirement for continuous data transfer between parties and for their continuous online availability.

In this section, we provide a more in-depth comparison and systematization of the SMPC protocols summarized in subsection 3.2. We discuss their strengths and weaknesses with the aim of outlining current challenges that need to be addressed. We look at the privacy and security guarantees provided by these protocols as well as the data type supported and the evaluation parameters. We also check whether the given SMPC protocols provide an OSI (see Table 6).

**GPU Utilization:** GPUs are considered one of the most significant foundations for the resurgence of ML, because their parallel architecture is well-suited to dense matrix operations. Consequently, ML frameworks, such as TensorFlow, PyTorch, and Caffe allow GPU acceleration. As GPUs played an important role in the success of modern ML techniques, they also became essential for scalable PPML. However, in the literature, most of the works on PPML are CPU-based and only two works consider the GPU-based research for PPML [39, 140]. One thing to remember is that while choosing and designing PPML protocols for the GPU, one must carefully calibrate them for the architecture. Protocols like Yao’s GC are less well suited for taking advantage of GPU parallelism compared to an SS-based protocol. Similarly, protocols that require extensive finite field arithmetic will incur more overhead on the GPU compared to the protocols that only rely on arithmetic modulo of a power of 2.

**Security:** Currently the fastest SMPC protocols only provide security with abort. This means that a malicious service provider will cause the computation to abort without any output thus rendering the later appearance of the input provider irrelevant. Some SMPC techniques provide fairness. Protocols providing security with abort or fairness will not suffice as in both cases an adversary can cause the protocol to abort, thus not producing the desired output for the user. This leads to denial of service and heavy economic losses for the service provider. Therefore, some SotA SMPC approaches,

as described in Table 6 ensure robustness, guaranteeing that the correct output is produced, no matter how the adversary behaves. **Secure Machine Learning (SML):** SML refers to preventing leakage of user information by protecting the process of ML. As can be seen in Table 6, different ways are used to achieve SML. In paper [17, 39, 103, 127], the authors used two-party computation, in papers [26, 83, 101, 118, 140, 146, 147], the authors used three-party computation, while in papers [38, 47, 84, 139] the authors used four-party computation. In Table 6, we also consider the input/output or model privacy that the above works aim to protect. As input privacy is the key feature of PPML, all the above works provide input privacy. However, only few works incorporate output and model privacy (broader and active area of research).

**Adversarial Model (AM):** AM defines the threats and adversary capabilities on a cryptographic protocol. It can be classified either based on the adversarial behavior or on the number of corruptions [93]. Based on their behavior, the adversaries are categorized into semi-honest and malicious. In a semi-honest setting, the adversary follows the protocol, but tries to glean additional information from the message. Most of the literature usually assumes a semi-honest adversary as mentioned in Table 3. Said adversary is limited in its offensive capabilities. This type of adversary model lowers the performance requirements. In malicious behavior, the adversary arbitrarily deviates from the protocol. This requires the adversary to either follow the protocol or do something completely different. The second one is based on the number of corruptions, which can be further classified into two categories: honest and dishonest majority. If  $N$  parties are taking part in SMPC, then in honest majority at most  $(\frac{N}{2} - 1)$  are allowed to be corrupt, ensuring that the number of honest parties is in the majority. Adversely, in dishonest majority parties are allowed to be corrupt only as high as  $N - 1$ .

**Data Types:** Any computable function can be securely evaluated in SMPC using two types of secret sharing: Additive Secret Sharing (ASS) [29] and Boolean Secret Sharing (BSS) [121, 148]. In ASS, the data is additively shared between the parties. For example,  $D$  is the original dataset, which is then shared to two parties: one party has the share  $D_1$ , while another party has the share  $D_2$ .  $D_1$  and  $D_2$  can reconstruct  $D$  by adding their data together. Therefore, as long as they do not collude, the original dataset  $D$  is kept private. ASS uses an arithmetic circuit and supports efficient calculations over integers and floating-point, that do not involve comparisons. In these protocols, additions are cheap, whereas multiplications are expensive. Alternatively, in BSS the data is XORed shared between the parties (bit by bit). For BSS, the SMPC uses boolean circuits and supports boolean points [121], as such they are well suited for comparison, division and multiplication operations. As mentioned in [21] to date, the majority of techniques (PPML) are implemented using integers, while there are undeniable limitations to integer arithmetic. In contrast, as can be seen in Table 6, the papers that we have surveyed for SMPC mostly use boolean points.

## 5 TAKEAWAYS

Following the results of our work, we have observed the following key takeaways, which may instigate further research:

- **High computational and communication costs.** The main reason HE has not been adopted for standardized use is related to

the high cost associated with model training. Since the amount of noise and the size of the ciphertext increase during mathematical operations on encrypted values, the training time required for an encrypted model increases almost exponentially with every processed batch. Consequently, though accuracy remains high enough to compare with plaintext models, the training time (while using HE), is multiple times longer than the plaintext counterpart. Other contributions rely on SMPC through distributed architectures, trading off the computational performance and communication overhead. SMPC offers a significantly better level of security at the expense of costly cryptographic operations, leading to computational and communication cost increases. For example, in SMPC each party has minimal computational costs. However communication between parties is required and this can cause increased communication costs. Both HE and SMPC need to modify the model structure to match the corresponding PPML protocols. This affects accuracy and hinders efficiency with existing frameworks. One can see that HE and SMPC have their own pros and cons in terms of security, effectiveness, efficiency and scalability. We believe that hybrid approaches to training and inference, such as the ones proposed by POSEIDON [130] can enjoy the benefits of each component, thus provisioning the optimal trade-off between ML task performance and privacy-preserving overhead.

- **Lack of open-source implementations.** While currently there is a gap between theoretical advances and real-world applications, there are certain open-source projects and tools dedicated to PPML [46, 69, 128]. OSI is the foundation of a lot of the modern scientific research – providing reliability, flexibility, transparency, and opportunities for collaboration. However, as can be seen in Table 4, OSI of HE protocols are few and far between as most implementations are private for various copyright or personal reasons. Also, as shown in Table 6, 8 of the 15 works provide OSIs for SMPC, while the remaining do not, but as stated in subsection 4.1, there are some other issues relating to protocol implementation, such as lack of proper documentation. Because of the sheer difficulty of the implementation task, most researchers only cover the results documented in the original paper, instead of recreating the implementations. This in turn causes problems, such as inability to reproduce the results of the papers and test the implementation in other environments or on different datasets. One of the main contributions of this work is to highlight the importance of OSI in PPML and the need for reproducibility and usability considerations in research. We aim to bridge the gap by encouraging researchers and practitioners to prioritize open-source practices and share their implementations as this would provide:
  - **Reproducibility and Progress:** OSI provide opportunity to other researchers to reproduce and validate the results of a paper. By sharing the code, researchers contribute to the transparency and integrity of the scientific process, allowing others to verify and build upon their work. This leads to increased confidence and fosters a culture of reproducibility. Additionally, OSI facilitate collaboration and knowledge sharing. When codes are made public, other researchers can build upon the existing work, enhance it, and develop new techniques more rapidly. This accelerates the progress in the field of PPML by leveraging the collective efforts and expertise of the research community.

- **Long-term sustainability and wider impact:** OSI are often maintained and supported by a community of contributors, ensuring their long-term sustainability. By encouraging researchers, to share their code, it promotes the continuous development, improvement, and maintenance of PPML implementations, thus addressing issues such as software bugs, compatibility with new platforms or libraries, and evolving security requirements, ensuring the longevity and usefulness of the implementations over time. Additionally, it lowers the barrier to entry for those interested in utilizing PPML for practical purposes.

- **Lack of possible attack analysis on implementations.** Every proposed implementation of HE covered here assumes that the server is a semi-honest third-party and can only listen and watch the results of the training and inference without attempting to gain any additional information through malicious means. As mentioned previously, the assumption is realistic in most real-world scenarios involving the use of cloud-based server providers. It does, however, reveal a lack of analysis when it comes to potential ML attacks on the protocols. As a result, the lack of analysis on potential attacks points to a possible gap in the security of HE, when used in PPML. That is due to the fact that ML tasks open new vectors of attack for malicious actors. Similarly, most of the SMPC protocols covered in this study consider the honest, but curious model, however, some studies also take the malicious model into account. While the works we investigated assume a non-colluding server, collusion between the parties must be taken into account in all of these adversary models. In SMPC, collusion is unavoidable and poses a severe privacy concern, because it allows parties to learn each other’s sensitive private input. Typically, parties in collusion share their data and function parameter settings with one another. Therefore, it is crucial to weight privacy against collusion, while creating an effective SMPC protocol.
- **Lack of privacy-conscious regulatory frameworks** The available privacy-related regulations may require companies to announce that they are collecting all data and possibly provide users with the choice to opt out of said data gathering. However this appears to be a zero-sum game. Privacy policies can help data owners determine which data is shared, under what conditions, with whom, and for what purposes. It is essential to ascertain whether the policy will be implemented on the client or server side, which would change it into a format, where the data owner could allow or restrict access to other users and the server for certain purposes (such as marketing) or at least remove access users posing a possible privacy threat.

## 6 CHALLENGES AND FUTURE DIRECTIONS

Despite the aforementioned techniques for protecting private data, while performing ML training, non-privacy ML algorithms are still frequently employed, and private data is still transferred to the cloud. To date, there is no silver-bullet technique, when it comes to achieving privacy in ML. The privacy degree offered by the techniques we presented varies a lot depending on many factors such as the ML algorithm used, the adversary’s capabilities and resources etc. Below we will discuss in detail the challenges of existing PPML techniques and their possible solutions.

While ML advancements are frequently offered, the PPML techniques covered in this article are linked to specific ML algorithms. Therefore, PPML techniques are required to cope with the most recent advancements in ML. This creates a challenge, where both HE and SMPC need to modify the model structure to match the corresponding PPML protocols. This affects the training and inference accuracy and hinders compatibility with existing ML frameworks.

HE and SMPC allow work on encrypted data, thereby preserving the utility of the original datasets. However, their domain is quite limited, and scalability can be a major issue due to high costs. Both methods have differing advantages and disadvantages and as can be seen in the covered papers, contributions, relying on SMPC through distributed architectures, trade off lower computational costs with higher communication overhead. SMPC offers a better level of security at the expense of costly networking operations, e.g., in SMPC each party has less computational cost compared to HE but requires a lot of communication between parties. This can lead to high communication costs. On the other hand, in HE, the server incurs a substantial computational cost, as it has to train the entire model on its own, at almost no communication cost, as all of the data needed for training is sent to the server once. A possible way to combat the high costs of both techniques and achieve a higher degree of privacy may require the combination of multiple PPML techniques. Recent literature has proposed combinations of FL with HE [43, 130], or DP with HE [75, 112] or SMPC [105, 157], aiming for higher privacy guarantees and lower costs.

This work also addresses the scarcity of OSI in PPML and makes concerted efforts to understand and overcome the underlying challenges. We investigate the reason behind this scarcity, including concerns related to security, compatibility, maintenance, and support. By raising awareness and emphasizing the benefits of OSI, we aim to encourage researchers to prioritize the sharing of PPML implementation. Additionally, we explore potential solutions and discuss the importance of collaboration, community building, and knowledge sharing to foster a culture of OSI in PPML.

In light of this, we conclude that these PPTs are still in a developmental phase and that, in the years to come, they will have developed to become an essential component of ML and cryptography. We conclude by summarizing our thoughts on potential future research directions involving PPTs that leverage and benefit multiple research communities such as ML, security, and privacy.

- As covered in the sections above, performing computation on encrypted data using HE is computationally expensive. More specifically, training ML models requires a lot of mathematical operations and performing them on encrypted data, when training ML models, raises the computation costs to an impractical level. One possible solution is to use a hybrid approach – HE and FE. The idea is to first encrypt the message using HE and then re-encrypt using symmetric FE. FE is used instead of HE for inner products and sum. This is comparatively faster and more efficient. After training, the message is first decrypted, following the same steps, with the FE symmetric key. It acquires the form of a HE ciphertext, and is further decrypted by use of the HE secret key with the aim of recovering the original plaintext.
- DNN have millions of parameters. This is the main reason, why they are computationally expensive. Additionally, training a DNN

on HE data increases the computing cost even more, sometimes to an unacceptable level. One possible approach for PPML is to use split learning (SL) [67], as it divides a DNN model so that part of the model is trained on the client side using plaintext data, and the remaining part is trained on the server side using encrypted data [78]. As part of the model is trained on plaintext, the DNN model’s overall computation costs are lowered. Also, the privacy of user data is preserved as *a*) SL itself is believed to be a promising approach for raw data protection, and *b*) the computation on the server side is performed on top of encrypted data, hence not revealing information about client data.

- FL and SL have offered solutions to the privacy problems in ML, however, they are not complete and present various security problems and privacy leakages [15, 91, 156]. HE can be used to solve the privacy leakage in SL and FL. Currently, there are two popular HE schemes that can be employed in ML, namely CKKS and TFHE, each with advantages and disadvantages. In the future, we would like to research the applications of these HE schemes in FL and SL and how they can be used in combination with each other as well as other privacy technologies in different scenarios.
- A great deal of work has recently been put into developing a reliable and secure protocol for ML tasks including SMPC. In SMPC, the parties constantly communicate to jointly compute a function, thus adding overhead. An alternative to SMPC is to employ a hybrid strategy that combines SL and Function Secret Sharing (FSS) [32]. The model’s initial layers are trained using SL on the client side, while its remaining layers are trained using FSS on the server side. Sending the data to the layers is all that is required to execute the layers on the client side. However, FSS is used to construct the secret shares, for each layer on the server.

## 7 CONCLUSION

As PPML has recently gained attention from both industry and academic researchers, in this work we provide a thorough analysis of SotA PPML based on HE and SMPC. First, we provide a general overview of the techniques used to implement privacy-preserving computation techniques on ML. We also describe various properties and settings pertaining to the privacy of ML models and data. This makes it possible to comprehend the entire range of PPML as well as the benefits and constraints of various sub-areas depending on the goals and settings. Second, we summarize the current SotA of PPML techniques and provide an analysis based on factors including the privacy goal, architecture, efficiency, and usability. With this method, we extracted recurring insights and flaws from the existing solutions. Third, we benchmarked and compared different PPML techniques, concerning privacy goals, communication, accuracy, and runtime to assess the viability of various PPML approaches for use in practical applications. Fourth, we list the key takeaways from our work, emphasizing the lessons learned and areas in need of further research effort. As such, this will help future researchers and practitioners apply PPML more effectively and have a better understanding of current limitations. Finally, we conclude that there are still many hurdles to overcome for HE and SMPC-based PPML to become practical. We outline the current challenges, that need to be addressed by the community, and discuss potential directions and approaches.

## ACKNOWLEDGMENTS

We wish to thank the reviewers of the paper for their invaluable comments, which helped shape and improve our work.

This work was funded by the HARPOCRATES EU research project (No. 101069535).

## REFERENCES

- [1] 2007. NVidia CUDA Toolkit. <https://developer.nvidia.com/cuda-toolkit>.
- [2] 2010. ImageNet Large Scale Visual Recognition Challenge (ILSVRC). <https://image-net.org/challenges/LSVRC/>.
- [3] 2011. Large Movie Review Dataset. <http://ai.stanford.edu/~amaas/data/sentiment/>.
- [4] 2012. ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). <https://image-net.org/challenges/LSVRC/2012/index.php>.
- [5] 2014. ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014). <https://image-net.org/challenges/LSVRC/2014/index.php>.
- [6] 2015. FV-NFLlib library. <https://github.com/CryptoExperts/FV-NFLlib>.
- [7] 2018. Microsoft SEAL Library. Online: <https://www.microsoft.com/en-us/research/project/microsoft-seal/>.
- [8] 2021. HELib Open-Source. <https://github.com/homenc/HELib>.
- [9] 2022. glmfit - Fit generalized linear regression model. <https://se.mathworks.com/help/stats/glmfit.html>
- [10] 2022. Lattigo v4: lattice-based multiparty homomorphic encryption library in Go. <https://github.com/tuneinsight/lattigo>.
- [11] 2022. OpenFHE. Online: <https://www.openfhe.org>.
- [12] 2022. Zama. Online: <https://www.zama.ai>.
- [13] 2022. Zama Concrete Framework. Online: <https://docs.zama.ai/tfhe-rs>.
- [14] 2022. Zama Concrete Framework. Online: <https://www.zama.ai/concrete-ml>.
- [15] Sharif Abuadba, Kyuyeon Kim, Minki Kim, Chandra Thapa, Seyit A Camtepe, Yansong Gao, Hyoungshick Kim, and Surya Nepal. 2020. Can we use split learning on 1d cnn models for privacy preserving training?. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*.
- [16] Abbas Acar, Hidayet Aksu, A Selcuk Ulugac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)* 51, 4 (2018), 1–35.
- [17] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. 2019. QUOTIENT: two-party secure neural network training and prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1231–1247.
- [18] Carlos Aguilar Melchor, Marc-Olivier Kilijian, Cédric Lefebvre, and Thomas Ricosset. 2018. A comparison of the homomorphic encryption libraries helib, seal and fv-nllib. In *International Conference on Security for Information Technology and Communications*. Springer, 425–442.
- [19] Ahmad Al Badawi, Louie Hoang, Chan Fook Mun, Kim Laine, and Khin Mi Mi Aung. 2020. Privft: Private and fast text classification with homomorphic encryption. *IEEE Access* 8 (2020), 226544–226556.
- [20] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*. Ieee, 1–6.
- [21] Mehrdad Aliasgari, Marina Blanton, Yihua Zhang, and Aaron Steele. 2013. Secure Computation on Floating Point Numbers. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society. <https://www.ndss-symposium.org/ndss2013/secure-computation-floating-point-numbers>
- [22] Tiago Almeida, Tiago Silva, Igor Santos, and Jose Maria Gomez Hidalgo. 2017. YouTube Spam Collection Data Set. <https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection>
- [23] Toshinori Araki, Assi Barak, Jun Furukawa, Tamar Lichter, Yehuda Lindell, Ariel Nof, Kazuma Ohara, Adi Watzman, and Or Weinstein. 2017. Optimized honest-majority MPC for malicious adversaries—breaking the 1 billion-gate per second barrier. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE.
- [24] Toshinori Araki, Assaf Barak, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. High-throughput secure three-party computation of kerberos ticket generation. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1841–1843.
- [25] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A Reuter, and Martin Strand. 2015. A guide to fully homomorphic encryption. *Cryptology ePrint Archive* (2015).
- [26] Nuttapon Attrapadung, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Takahiro Matsuda, Ibuki Mishina, Hiraku Morita, and Jacob CN Schuldt. 2022. Adam in Private: Secure and Fast Training of Deep Neural Networks with Adaptive Moment Estimation. *Proceedings on Privacy Enhancing Technologies* 4 (2022), 746–767.
- [27] Alexandros Bakas, Antonis Michalas, and Tassos Dimitriou. 2022. Private lives matter: a differential private functional encryption scheme. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*. 300–311.
- [28] Elaine Barker, Elaine Barker, William Burr, William Polk, Miles Smid, et al. 2006. *Recommendation for key management: Part 1: General*. National Institute of Standards and Technology, Technology Administration . . .
- [29] George Robert Blakley. 1979. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*. IEEE Computer Society.
- [30] Charlotte Bonte and Frederik Vercauteren. 2018. Privacy-preserving logistic regression training. *BMC medical genomics* 11, 4 (2018), 13–21.
- [31] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. 2020. Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. *Journal of Mathematical Cryptology* 14, 1 (2020), 316–338.
- [32] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2016. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1292–1303.
- [33] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)* 6, 3 (2014), 1–36.
- [34] Megha Byali, Arun Joseph, Arpita Patra, and Divya Ravi. 2018. Fast secure computation for small population over the internet. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 677–694.
- [35] José Cabrero-Holgueras and Sergio Pastrana. 2021. SoK: Privacy-preserving computation techniques for deep learning. *Proceedings on Privacy Enhancing Technologies* 2021, 4 (2021), 139–162.
- [36] Mahawaga Arachchige Pathum Chamikara, Peter Bertok, Ibrahim Khalil, Dongxi Liu, and Seyit Camtepe. 2021. Privacy preserving distributed machine learning with federated learning. *Computer Communications* 171 (2021).
- [37] Konstantinos Chatzilygeroudis, Ioannis Hatzilygeroudis, and Isidoros Perikos. 2021. Machine learning basics. In *Intelligent Computing for Interactive System Design: Statistics, Digital Signal Processing, and Machine Learning in Practice*.
- [38] Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. 2020. Trident: Efficient 4pc framework for privacy preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society.
- [39] Zheng Chen, Feng Zhang, Amelie Chi Zhou, Jidong Zhai, Chenyang Zhang, and Xiaoyong Du. 2020. ParSecureML: An efficient parallel secure machine learning framework on GPUs. In *49th International Conference on Parallel Processing*.
- [40] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 409–437.
- [41] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. 2016. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*. Springer, 3–33.
- [42] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology* 33, 1 (2020), 34–91.
- [43] Christopher A Choquette-Choo, Natalie Dullerud, Adam Dzedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. 2021. Capc learning: Confidential and private collaborative learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. [https://openreview.net/forum?id=h2Ebj4\\_wMvQ](https://openreview.net/forum?id=h2Ebj4_wMvQ)
- [44] Josh D Cohen and Michael J Fischer. 1985. *A robust and verifiable cryptographically secure election scheme*. Yale University. Department of Computer Science.
- [45] DBpedia Community. 2021. Ontology (DBO). <https://www.dbpedia.org/resources/ontology/>
- [46] Morten Dahl, Jason Mancuso, Yann Dupis, Ben Decoste, Morgan Giraud, Ian Livingstone, Justin Patriquin, and Gavin Uhma. 2018. Private machine learning in tensorflow using secure computation. (2018).
- [47] Anders Dalskov, Daniel Escudero, and Marcel Keller. 2021. Fantastic Four: {Honest-Majority} {Four-Party} Secure Computation With Malicious Security. In *30th USENIX Security Symposium (USENIX Security 21)*. 2183–2200.
- [48] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. Fully homomorphic encryption over the integers. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer.
- [49] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml>
- [50] Léo Ducas and Daniele Micciancio. 2015. FHEW: bootstrapping homomorphic encryption in less than a second. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 617–640.
- [51] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006*. Proceedings 3. Springer, 265–284.
- [52] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985).
- [53] Saroja Erabelli. 2020. *pyFHE-a Python library for fully homomorphic encryption*. Ph. D. Dissertation. Massachusetts Institute of Technology.

- [54] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. 2018. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security* 2, 2-3 (2018), 70–246.
- [55] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012).
- [56] Ronald Aylmer Fisher. 1988. Iris Data Set. <https://archive.ics.uci.edu/ml/datasets/Iris>
- [57] Adrià Gascón, Philipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. 2016. Secure Linear Regression on Vertically Partitioned Datasets. *IACR Cryptol. ePrint Arch.* 2016 (2016), 892.
- [58] Dong-yuan Ge, Xi-fan Yao, Wen-jiang Xiang, Xue-jun Wen, and En-chen Liu. 2019. Design of High Accuracy Detector for MNIST Handwritten Digit Recognition Based on Convolutional Neural Network. In *2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA)*.
- [59] Craig Gentry. 2009. A fully homomorphic encryption scheme. Stanford university.
- [60] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 169–178.
- [61] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*. Springer, 75–92.
- [62] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*. PMLR, 201–210.
- [63] Oded Goldreich. 1998. Secure multi-party computation. *Manuscript. Preliminary version* 78, 110 (1998).
- [64] O Goldreich, S Micali, and A Wigderson. 1987. A Completeness Theorem for Protocols with Honest Majority. In *STOC* 87.
- [65] S Dov Gordon, Samuel Ranellucci, and Xiao Wang. 2018. Secure computation with low communication from cross-checking. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer.
- [66] Antonio Gulli and Paolo Ferragina. 2005. [http://groups.di.unipi.it/~gulli/AG\\_corpus\\_of\\_news\\_articles.html](http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html)
- [67] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (2018), 1–8.
- [68] Brett Hemenway, Steve Lu, Rafail Ostrovsky, and William Welser Iv. 2016. High-precision secure computation of satellite collision probabilities. In *International Conference on Security and Cryptography for Networks*. Springer, 169–187.
- [69] Wilko Henecka, Stefan K ögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. 2010. TASTY: tool for automating secure two-party computations. In *Proceedings of the 17th ACM conference on Computer and communications security*. 451–462.
- [70] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. 2018. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 123–142.
- [71] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 603–618.
- [72] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15–20, 2018, Volume 1: Long Papers*. Association for Computational Linguistics, 328–339. <https://doi.org/10.18653/v1/P18-1031>
- [73] Yelp Inc. 2018. Yelp dataset. <https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset>
- [74] Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. 2015. Secure computation with minimal interaction, revisited. In *Annual Cryptology Conference*. Springer, 359–378.
- [75] Bin Jia, Xiaosong Zhang, Jiewen Liu, Yang Zhang, Ke Huang, and Yongquan Liang. 2022. Blockchain-Enabled Federated Learning Data Protection Aggregation Scheme With Differential Privacy and Homomorphic Encryption in IIoT. *IEEE Transactions on Industrial Informatics* 18, 6 (2022), 4049–4058. <https://doi.org/10.1109/TII.2021.3085960>
- [76] Tanveer Khan, Alexandros Bakas, and Antonis Michalas. 2021. Blind faith: Privacy-preserving machine learning using function approximation. In *2021 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1–7.
- [77] Tanveer Khan, Khoa Nguyen, and Antonis Michalas. 2023. A More Secure Split: Enhancing the Security of Privacy-Preserving Split Learning. In *Nordic Conference on Secure IT Systems*. Springer, 307–329.
- [78] Tanveer Khan, Khoa Nguyen, and Antonis Michalas. 2023. Split Ways: Privacy-Preserving Training of Encrypted Data Using Split Learning. In *2023 Workshops of the EDBT/ICDT Joint Conference, EDBT/ICDT-WS 2023, 28 March 2023*. CEUR-WS.
- [79] Memoona Khanum, Tahira Mahboob, Warda Imtiaz, Humaraia Abdul Ghafoor, and Rabeea Sehar. 2015. A survey on unsupervised machine learning algorithms for automation, classification and maintenance. *International Journal of Computer Applications* 119, 13 (2015).
- [80] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. 2002. *Logistic regression*. Springer.
- [81] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. 2021. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems* 34 (2021).
- [82] Amina Bel Korchi and Nadia El Mrabet. 2019. A practical use case of homomorphic encryption. In *2019 International Conference on Cyberworlds (CW)*. IEEE.
- [83] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. 2021. {SWIFT}: Super-fast and Robust {Privacy-Preserving} Machine Learning. In *30th USENIX Security Symposium (USENIX Security 21)*. 2651–2668.
- [84] Nishat Koti, Arpita Patra, Rahul Rachuri, and Ajith Suresh. 2022. Tetrad: Actively Secure 4PC for Secure Training and Inference. In *Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, 24 - 28 April, 2022*. The Internet Society.
- [85] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).
- [87] Ya Le and Xuan Yang. 2015. Tiny imagenet visual recognition challenge. *CS 231N* 7, 7 (2015), 3.
- [88] Yann LeCun. 1998. The MNIST database of handwritten digits. <http://yann.lecun.com/exdb/mnist/> (1998).
- [89] Yann LeCun et al. 2015. LeNet-5, convolutional neural networks. *URL: http://yann.lecun.com/exdb/lenet* 20, 5 (2015), 14.
- [90] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998).
- [91] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2022. Label leakage and protection in two-party split learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022*.
- [92] Yehida Lindell. 2005. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*. IGI global.
- [93] Yehuda Lindell. 2020. Secure multiparty computation. *Commun. ACM* 64, 1 (2020), 86–96.
- [94] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. 2017. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 619–631.
- [95] Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. 2020. Glyph: Fast and accurately training deep neural networks on encrypted data. *Advances in Neural Information Processing Systems* 33 (2020), 9193–9202.
- [96] Donghang Lu, Albert Yu, Aniket Kate, and Hemanta Maji. 2022. Polymath: Low-Latency MPC via Secure Polynomial Evaluations and Its Applications. *Proceedings on Privacy Enhancing Technologies* 1 (2022), 396–416.
- [97] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 1–23.
- [98] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA. <http://www.aclweb.org/anthology/P11-1015>
- [99] Eleftheria Makri, Dragos Rotaru, Frederik Vercauteren, and Sameer Wagh. 2021. Rabbit: Efficient comparison for secure multi-party computation. In *International Conference on Financial Cryptography and Data Security*. Springer, 249–270.
- [100] Kentaro Mihara, Ryohei Yamaguchi, Miguel Mitsuishi, and Yusuke Maruyama. 2020. Neural Network Training With Homomorphic Encryption. *CoRR* abs/2012.13552 (2020). arXiv:2012.13552 <https://arxiv.org/abs/2012.13552>
- [101] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 35–52.
- [102] Payman Mohassel, Mike Rosulek, and Ye Zhang. 2015. Fast and secure three-party computation: The garbled circuit approach. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 591–602.
- [103] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 19–38.
- [104] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. 2021. Multiparty Homomorphic Encryption from Ring-Learning-with-Errors. *Proceedings on Privacy Enhancing Technologies* 2021, 4 (2021), 291–311. <https://doi.org/doi:10.2478/popets-2021-0071>

- [105] Vaikkunth Mugunthan, Antigoni Polychroniadou, David Byrd, and Tucker Hybinette Balch. 2019. Smpai: Secure multi-party computation for federated learning. In *Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services*.
- [106] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 113–124.
- [107] Karthik Nandakumar, Nalini Ratha, Sharath Pankanti, and Shai Halevi. 2019. Towards deep neural network training on encrypted data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.
- [108] Lucien KL Ng and Sherman SM Chow. 2023. SoK: Cryptographic Neural-Network Computation. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 497–514.
- [109] K Nguyen, T Khan, and A Michalas. 2023. Split Without a Leak: Reducing Privacy Leakage in Split Learning. In *19th EAI International Conference on Security and Privacy in Communication Networks (SecureComm'23)*. Springer.
- [110] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. 2013. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE symposium on security and privacy*. IEEE, 334–348.
- [111] Daniel Olszewski, Allison Lu, Carson Stillman, Kevin Warren, Cole Kitroser, Alejandro Pascual, Divyjayoti Ukirde, Kevin Butler, and Patrick Traynor. 2023. "Get in Researchers; We're Measuring Reproducibility": A Reproducibility Study of Machine Learning Papers in Tier 1 Security Conferences. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 3433–3459.
- [112] Kwabena Owusu-Agyemeng, Zhen Qin, Hu Xiong, Yao Liu, Tianming Zhuang, and Zhiguang Qin. 2021. MSDP: multi-scheme privacy-preserving deep learning via differential privacy. *Personal and Ubiquitous Computing* (2021), 1–13.
- [113] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*. Springer, 223–238.
- [114] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. 2018. SoK: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 399–414.
- [115] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. 2021. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2113–2129.
- [116] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
- [117] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2021. {ABY2.0}: Improved {Mixed-Protocol} Secure {Two-Party} Computation. In *30th USENIX Security Symposium (USENIX Security 21)*. 2165–2182.
- [118] Arpita Patra and Ajith Suresh. 2020. BLAZE: blazing fast privacy-preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society. <https://www.ndss-symposium.org/ndss-paper/blaze-blazing-fast-privacy-preserving-machine-learning/>
- [119] Pedro Silveira Pisa, Michel Abdalla, and Otto Carlos Muniz Bandeira Duarte. 2012. Somewhat homomorphic encryption scheme for arithmetic operations on large integers. In *2012 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 1–8.
- [120] Manish M. Potey, C.A. Dhote, and Deepak H. Sharma. 2016. Homomorphic Encryption for Security of Cloud Data. *Procedia Computer Science* 79 (2016), 175–181. <https://doi.org/10.1016/j.procs.2016.03.023> Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016.
- [121] Pille Pullonen and Sander Siim. 2015. Combining secret sharing and garbled circuits for efficient private IEEE 754 floating-point computations. In *International Conference on Financial Cryptography and Data Security*. Springer, 172–183.
- [122] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)* 56, 6 (2009), 1–40.
- [123] Oded Regev. 2010. The learning with errors problem. *Invited survey in CCC* 7, 30 (2010), 11.
- [124] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. 1978. On data banks and privacy homomorphisms. *Foundations of secure computation* 4, 11 (1978).
- [125] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. 1983. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 26, 1 (1983), 96–99.
- [126] Bitá Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. 2018. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th annual design automation conference*. 1–6.
- [127] Théo Ryffel, Pierre Tholoniat, David Pointcheval, and Francis Bach. 2020. Ariann: Low-interaction privacy-preserving deep learning via function secret sharing. *Proceedings on Privacy Enhancing Technologies* 2022, 1 (2020), 291–316.
- [128] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. In *Privacy Preserving Machine Learning, NeurIPS 2018 Workshop*. NeurIPS.
- [129] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. IEEE, 57–64.
- [130] Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. 2021. POSEIDON: privacy-preserving federated neural network learning. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society.
- [131] George AF Seber and Alan J Lee. 2012. *Linear regression analysis*. John Wiley & Sons.
- [132] G Shafi and M Silvio. 1982. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*. 365–77.
- [133] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
- [134] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.
- [135] Sharma Shreya. 2019. Secure-ML Implementation. Online: <https://github.com/shreya-28/Secure-ML>.
- [136] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- [137] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. 2016. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. Ieee, 1310–1315.
- [138] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification?. In *China national conference on Chinese computational linguistics*. Springer, 194–206.
- [139] Ajith Suresh. 2021. MPCLeague: Robust MPC Platform for Privacy-Preserving Machine Learning. In *Distributed and Private Machine Learning (DPML) ICLR Workshop*. International Conference on Learning Representations (ICLR).
- [140] Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. 2021. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1021–1038.
- [141] Haixu Tang, XiaoFeng Wang, Shuang Wang, and Xiaoqian Jiang. 2018. Genomic data privacy and security protection competition. <http://www.humangenomeprivacy.org/2017>
- [142] Han Tian, Chaoliang Zeng, Zhenghang Ren, Di Chai, Junxue Zhang, Kai Chen, and Qiang Yang. 2022. Sphinx: Enabling privacy-preserving online learning over the cloud. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2487–2501.
- [143] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 1–11.
- [144] Mayank Varia. 2018. Cryptographically Secure Data Analysis for Social Good. In *Enigma 2018 (Enigma 2018)*. USENIX Association, Santa Clara, CA. <https://www.usenix.org/node/208174>
- [145] Alexander Viand, Patrick Jattke, and Anwar Hithnawi. 2021. SoK: Fully homomorphic encryption compilers. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1092–1108.
- [146] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proc. Priv. Enhancing Technol.* 2019, 3 (2019), 26–49.
- [147] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2021. Falcon: Honest-Majority Maliciously Secure Framework for Private Deep Learning. *Proceedings on Privacy Enhancing Technologies* 2021, 1 (2021), 188–208.
- [148] Daoshun Wang, Lei Zhang, Ning Ma, and Xiaobo Li. 2007. Two secret sharing schemes based on Boolean operations. *Pattern Recognition* 40, 10 (2007).
- [149] Jean-Luc Watson, Sameer Wagh, and Raluca Ada Popa. 2022. Piranha: A {GPU} platform for secure computation. In *31st USENIX Security Symposium (USENIX Security 22)*. 827–844.
- [150] Marco A Wiering and Martijn Van Otterlo. 2012. Reinforcement learning. *Adaptation, learning, and optimization* 12, 3 (2012), 729.
- [151] Alexander Wood, Kayvan Najarian, and Delaram Kahrobaei. 2020. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Computing Surveys (CSUR)* 53, 4 (2020), 1–35.
- [152] Runhua Xu, Nathalie Baracaldo, and James Joshi. 2021. Privacy-preserving machine learning: Methods, challenges and directions. *CoRR abs/2108.04417* (2021). arXiv:2108.04417 <https://arxiv.org/abs/2108.04417>
- [153] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).
- [154] Andrew Chi-Chih Yao. 1982. Protocols for secure computations. *23rd Annual Symposium on Foundations of Computer Science (sfcS 1982)* (1982), 160–164.

- [155] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE.
- [156] Xuefei Yin, Yanming Zhu, and Jiankun Hu. 2021. A Comprehensive Survey of Privacy-Preserving Federated Learning: A Taxonomy, Review, and Future Directions. *Comput. Surveys* 54, 6, Article 131 (jul 2021), 36 pages.
- [157] Sen Yuan, Milan Shen, Ilya Mironov, and Anderson Nascimento. 2021. Label Private Deep Learning Training based on Secure Multiparty Computation and Differential Privacy. In *NeurIPS 2021 Workshop Privacy in Machine Learning*.
- [158] Xiang Zhang. 2015. [https://huggingface.co/datasets/dbpedia\\_14](https://huggingface.co/datasets/dbpedia_14)
- [159] Xiang Zhang. 2020. AG News Classification Dataset. <https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset>
- [160] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015).
- [161] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. 2019. Secure multi-party computation: theory, practice and applications. *Information Sciences* 476 (2019), 357–372.
- [162] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. 2021. Cerebro: A Platform for {Multi-Party} Cryptographic Collaborative Learning. In *30th USENIX Security Symposium (USENIX'21)*.
- [163] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).

## A PRELIMINARIES

In this section, we discuss the categorization, construction, and schemes of HE. Additionally, we present the basic primitives developed for constructing SMPC protocols with different properties, security notations, and dimensions. We conclude this section by discussing the various ML models and datasets that PPML employs<sup>5</sup>.

### A.1 Homomorphic Encryption

There are four main categories of HE based on the scheme’s ability to perform various mathematical calculations. The main categories are: *Partially Homomorphic Encryption (PHE)* [44, 52, 113, 125, 132], *Fully Homomorphic Encryption (FHE)* [41, 42, 53, 59, 145], *Somewhat Homomorphic Encryption (SHE)* [25, 48] and *Leveled Homomorphic Encryption (LHE)* [25, 33, 40, 61].

FHE schemes have the highest potential in various applications. However, their use and implementation is heavily stunted by the complex mathematics required to implement them [16]. FHE’s mathematics require a different approach to encryption and decryption compared to the ones provided in prior PHE schemes, such as RSA or Elgamal. The major breakthrough that was in order in the field of HE occurred when Gentry proved that ciphertext homomorphism can be achieved through the use of lattices [59]. The lattice-based cryptography in Gentry’s work used the security provided in the Learning with Error (LWE) problem described by Oded Regev [123]. Future schemes of HE continued to rely on the high security provided by the LWE assumption and its variations (DLWE [122], ring LWE [97], etc). However, because of some major drawbacks in FHE, researchers, including Gentry himself, focused on alternative solutions. This led to the SHE and LHE schemes mentioned above, which, though based on the same concept, lowered the complexity of the algorithms by limiting the depth of the circuit through scheme parameters or a predefined value to avoid bootstrapping [16, 33, 35, 106].

Generally, a HE scheme can be defined as [60]:

*Definition A.1 (Homomorphic Encryption).* Let HE be a (public-key) homomorphic encryption scheme with a quadruple of PPT algorithms  $HE = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval})$  such that:

- **HE.KeyGen** : The key generation algorithm  $(pk, evk, sk) \leftarrow \text{HE.KeyGen}(1^\lambda)$  takes as input a unary representation of the security parameter  $\lambda$ , and outputs a public key  $pk$ , an evaluation key  $evk$  and a private key  $sk$ .
- **HE.Enc** : The encryption algorithm  $c \leftarrow \text{HE.Enc}(pk, x)$  takes as input the public key  $pk$  and a message  $x$  and outputs a ciphertext  $c$ .
- **HE.Eval** : The evaluation algorithm  $c_f \leftarrow \text{HE.Eval}(evk, f, c_1, \dots, c_n)$  takes as input the evaluation key  $evk$ , a function  $f$ , and a set of  $n$  ciphertexts, and outputs a ciphertext  $c_f$ .
- **HE.Dec** : The decryption algorithm  $\text{HE.Dec}(sk, c) \rightarrow x$ , takes as input the secret key  $sk$  and a ciphertext  $c$ , and outputs a plaintext  $x$ .

*Limitation:* Despite advancements over time, all HE techniques still require high-end machines to properly implement bootstrapping and are significantly more computationally expensive when compared to other privacy-preserving alternatives. Furthermore, SHE and LHE schemes limit the number of operations performed on a ciphertext, which might result in lower accuracy and reduced functionality in certain applications.

**HE schemes:** HE includes a variety of encryption schemes that can perform various types of computations over encrypted data. These encryption schemes define, which operations are available and the type of activation and architectures that can be used. The common encryption schemes BGV [33], and FV [55] (see Figure 2) share many similarities such as Single Instruction Multiple Data (SIMD) operations with an integer-only message space and support bootstrapping [145]. The two other popular schemes, CKKS [40] and TFHE [41], support bootstrapping (which is faster in TFHE). The CKKS scheme uses real numbers as the message space and TFHE only supports single bits in the message space. However, the fact that TFHE does not support SIMD operations, makes it less attractive for matrix multiplication. Another major HE scheme, not used by any of the papers, is FHEW [50]. FHEW is an FHE scheme, which allows homomorphic NAND operations on two encrypted bits  $E(b_1)$  and  $E(b_2)$  to get the resulting bit  $E(b_1 \wedge b_2)$ . As the scheme is primarily built for only NAND operations, it has limited uses in more complex applications. However, the main benefit of FHEW is that it allows much faster bootstrapping, when compared to other HE schemes.

### A.2 Secure Multi-party Computation

To properly apply SMPC techniques to ML, it is important to manage the costs of the SMPC frameworks, as ML requires numerous rounds of communication between the participant and the server. This may lead to significant communication costs. As such, various techniques have been developed over the years to help construct cost-effective SMPC frameworks, such as:

**Secret Sharing (SS):** Splitting a secret into multiple shares and distributing each share to a different party. The secret can only be recovered when all parties join shares [133].

*Limitation:* In SS, multiple parties use a protocol to jointly compute a function on their inputs. As SS requires communication between parties, it can lead to high communication cost.

<sup>5</sup>Due to space constraints, “Machine Learning and Datasets” are in appendix section.

**Table 2: Comparison among privacy-preserving training methods using HE. All of the listed works assume a semi-honest threat model and do not take into account a potentially malicious server or client.**

| Framework | Year                | Convolutional       |            |         | CKKS | FV | TFHE | BGV        | SHE | LHE | FHE | MHE                | MNIST | CIFAR | UCI | iDASH |
|-----------|---------------------|---------------------|------------|---------|------|----|------|------------|-----|-----|-----|--------------------|-------|-------|-----|-------|
|           |                     | Supported Layers    | Non-linear | Pooling |      |    |      |            |     |     |     |                    |       |       |     |       |
|           |                     | Theoretical Metrics |            |         |      |    |      | HE Setting |     |     |     | Evaluation Dataset |       |       |     |       |
|           |                     | Theoretical Metrics |            |         |      |    |      | HE Setting |     |     |     | Evaluation Metrics |       |       |     |       |
| HE        | Bonte & Vercauteren | 2018                | ✗          | ✓       | ✗    | ✗  | ✓    | ✗          | ✓   | ✗   | ✗   | ✗                  | ✗     | ✗     | ✗   | ✓     |
|           | CryptoDL            | 2019                | ✓          | ✓       | ✓    | ✗  | ✗    | ✓          | ✓   | ✗   | ✗   | ✗                  | ✓     | ✓     | ✓   | ✗     |
|           | Nandakumar et al.   | 2019                | ✗          | ✓       | ✗    | ✗  | ✗    | ✓          | ✗   | ✗   | ✓   | ✗                  | ✓     | ✗     | ✓   | ✗     |
|           | PrivFT              | 2019                | ✗          | ✗       | ✗    | ✓  | ✗    | ✗          | ✗   | ✗   | ✓   | ✗                  | ✗     | ✗     | ✓   | ✗     |
|           | Mihara et al.       | 2020                | ✗          | ✓       | ✗    | ✓  | ✗    | ✗          | ✗   | ✗   | ✓   | ✗                  | ✗     | ✗     | ✓   | ✗     |
|           | Sphinx              | 2022                | ✓          | ✓       | ✓    | ✓  | ✗    | ✗          | ✗   | ✗   | ✓   | ✗                  | ✗     | ✓     | ✗   | ✗     |
| Hybrid    | Glyph               | 2020                | ✓          | ✓       | ✓    | ✗  | ✗    | ✓          | ✓   | ✓   | ✗   | ✗                  | ✓     | ✗     | ✗   |       |
|           | POSEIDON            | 2021                | ✓          | ✓       | ✓    | ✓  | ✗    | ✗          | ✗   | ✗   | ✗   | ✓                  | ✓     | ✗     | ✗   |       |

**Table 3: Comparison between privacy-preserving training methods using SMPC**

| Framework | Year             | Threat Model        | Convolutional    |      |         |        | OT | GC                 | Secret Sharing | FSS | LAN | WAN                          | MNIST | CIFAR | Tiny ImageNet | ImageNet | From SecureML | From MiniONN | LeNet | AlexNet | VGG16 |
|-----------|------------------|---------------------|------------------|------|---------|--------|----|--------------------|----------------|-----|-----|------------------------------|-------|-------|---------------|----------|---------------|--------------|-------|---------|-------|
|           |                  |                     | Supported Layers | ReLU | Maxpool | Linear |    |                    |                |     |     |                              |       |       |               |          |               |              |       |         |       |
|           |                  | Theoretical Metrics |                  |      |         |        |    | Evaluation Dataset |                |     |     | Neural Network Architectures |       |       |               |          |               |              |       |         |       |
|           |                  | Theoretical Metrics |                  |      |         |        |    | Evaluation Metrics |                |     |     | Neural Network Architectures |       |       |               |          |               |              |       |         |       |
| 2PC       | SecureML         | 2017                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✗             | ✗            | ✗     | ✗       |       |
|           | Quotient         | 2019                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✗             | ✗            | ✗     | ✗       |       |
|           | ParSecureML      | 2020                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✗             | ✗            | ✗     | ✗       |       |
|           | ABY2             | 2021                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✗             | ✗            | ✗     | ✗       |       |
|           | AriaNN           | 2022                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✓                            | ✗     | ✗     | ✓             | ✓        | ✗             | ✗            | ✓     | ✓       |       |
| 3PC       | ABY <sup>3</sup> | 2018                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✗            | ✗     | ✗       |       |
|           | SecureNN         | 2019                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✗            | ✗     | ✗       |       |
|           | BLAZE            | 2020                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✗            | ✗     | ✗       |       |
|           | Falcon           | 2020                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✓                            | ✓     | ✓     | ✓             | ✓        | ✓             | ✓            | ✓     | ✓       |       |
|           | SWIFT            | 2021                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✓                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✓            | ✓     | ✓       |       |
|           | Adam in Private  | 2021                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✓                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✓            | ✓     | ✓       |       |
|           | CryptGPU         | 2021                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✓                            | ✓     | ✓     | ✓             | ✓        | ✓             | ✓            | ✓     | ✓       |       |
| Piranha   | 2022             | ●                   | ✓                | ✓    | ✓       | ✓      | ✓  | ✓                  | ●              | ✓   | ✓   | ✗                            | ✗     | ✗     | ✓             | ✓        | ✓             | ✓            | ✓     |         |       |
| 4PC       | Trident          | 2020                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✓            | ✓     | ✓       |       |
|           | Fantastic Four   | 2021                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✗                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✓            | ✓     | ✓       |       |
|           | MPCLeague        | 2021                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✗              | ●   | ✓   | ✓                            | ✓     | ✓     | ✓             | ✓        | ✓             | ✓            | ✓     | ✓       |       |
|           | Tetrad           | 2021                | ●                | ✓    | ✓       | ✓      | ✓  | ✓                  | ✓              | ●   | ✓   | ✓                            | ✗     | ✗     | ✗             | ✓        | ✓             | ✓            | ✓     | ✓       |       |

**Table 4: Comparison of training time, accuracy and details for privacy-preserving training methods using HE. \* - None of the listed papers provide open-source code for their implementation.**

| Framework | Year                | Training time(h) | Training accuracy(%) | CPU         | GPU                   | RAM(GB)    | Environment | Algorithms         | Availability* |
|-----------|---------------------|------------------|----------------------|-------------|-----------------------|------------|-------------|--------------------|---------------|
|           |                     |                  |                      |             |                       |            |             |                    |               |
| HE        | Bonte & Vercauteren | 2018             | 0.367-0.75           | 62.98-94.16 | -                     | -          | -           | ✓                  |               |
|           | CryptoDL            | 2019             | 2.91                 | 99.0        | 12-core CPU           | -          | 48          | Ubuntu 14.04 (VM)  | ✓             |
|           | Nandakumar et al.   | 2019             | 60K-846K             | 96.4-97.8   | Intel Xeon E5-2698 v3 | -          | 250         | Linux              | ✗             |
|           | PrivFT              | 2019             | 120.96               | 91.49-98.80 | Intel Xeon E5-2620    | DGX-1 V100 | 180         | ArchLinux (CPU)    | ✓             |
|           | Mihara et al.       | 2020             | 29.8                 | 98.47       | Intel Core i7-8700K   | -          | 32          | -                  | ✓             |
|           | Sphinx              | 2022             | 1560                 | 72.0-96.0   | Intel Xeon E5-2683 v4 | -          | 128         | Ubuntu 18.04.5 LTS | ✓             |
| Hybrid    | Glyph               | 2020             | 192                  | 98.6        | Intel Xeon E7-8890 v4 | -          | 256         | -                  | ✗             |
|           | POSEIDON            | 2021             | 1.47                 | 80.2-96.9   | Intel Xeon E5-2680 v3 | -          | 256         | Linux              | ✓             |

**Table 5: Comparison of training time and communication overhead, while training various frameworks over popular benchmarking network architectures in the PPML domain on the MNIST dataset. All networks and frameworks are exposed to a 15 epoch-training. Network 1 is a 3-layered fully-connected network from SecureML [103], Network 2 is a 4-layered network from MiniONN [94] that contains 2 convolution layers and 2 fully connected layers. Network 3 is LeNet [90] with 2 convolution layers and 2 fully connected layers.**

| Number of parties | Framework | Network 1 (SecureML) |           |          | Network 2 (MiniONN) |           |          | LeNet      |             |          |
|-------------------|-----------|----------------------|-----------|----------|---------------------|-----------|----------|------------|-------------|----------|
|                   |           | Local host           |           |          | Local host          |           |          | Local host |             |          |
|                   |           | Time (sec)           | Comm (MB) | Accuracy | Time (sec)          | Comm (MB) | Accuracy | Time (sec) | Comm (MB)   | Accuracy |
| 2PC               | SecureML  | -                    | -         | 92.98    | -                   | -         | -        | -          | -           | -        |
| 3PC               | SecureNN  | -                    | -         | -        | -                   | -         | -        | 136.553    | 21058.3     | -        |
| 3PC               | Falcon    | 2.38                 | 256.57    | -        | 67.75               | 6944.4    | -        | 173.9      | 1800        | -        |
| 2PC               | AriaNN    | 5.38 (h)             | 38757.89  | 97.96    | 12.81 (h)           | 868308.48 | 97.92    | 14.15 (h)  | 1324597.248 | 89.36    |

**Table 6: Comparison among privacy-preserving training methods using SMPC**

| Framework | Model            | Input/Output        | GOD | Fairness | Abort | Integer | Boolean | Floating Point | Public | Private | Accuracy | Complexity | Communication | Time |                |            |                |                           |
|-----------|------------------|---------------------|-----|----------|-------|---------|---------|----------------|--------|---------|----------|------------|---------------|------|----------------|------------|----------------|---------------------------|
|           |                  | Privacy             |     |          |       |         |         |                |        |         |          |            |               |      | Security       | Data types | Implementation | Parameters for Comparison |
|           |                  | Theoretical Metrics |     |          |       |         |         |                |        |         |          |            |               |      | Implementation |            |                |                           |
| 2PC       | SecureML         | ●                   | -   | -        | -     | -       | ✓       | ✓              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | Quotient         | ●                   | -   | -        | -     | ×       | ✓       | ×              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | ParSecureML      | ●                   | -   | -        | -     | -       | -       | -              | ●      | ●       | -        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | ABY <sup>2</sup> | ●                   | -   | -        | -     | ✓       | ✓       | -              | ●      | ●       | -        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | AriaNN           | ●                   | -   | -        | -     | -       | ✓       | -              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
| 3PC       | SecureNN         | ●                   | ×   | ×        | ✓     | ×       | ✓       | ×              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | ABY <sup>3</sup> | ●                   | ×   | ×        | ✓     | ×       | ✓       | ×              | ●      | ●       | ✓        | -          | ✓             | ✓    |                |            |                |                           |
|           | BLAZE            | ●                   | ×   | ✓        | ×     | ✓       | ✓       | -              | ●      | ●       | ×        | ✓          | ✓             | ×    |                |            |                |                           |
|           | Falcon           | ●                   | ×   | ×        | ✓     | ✓       | -       | -              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | SWIFT            | ●                   | ✓   | ×        | ×     | -       | ✓       | -              | ●      | ●       | ×        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | Adam in Private  | ●                   | ×   | ×        | ✓     | ✓       | ✓       | ✓              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | CryptGPU         | ●                   | -   | -        | -     | -       | ×       | ✓              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | Piranha          | ●                   | -   | -        | -     | -       | -       | -              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
| 4PC       | Trident          | ●                   | ×   | ✓        | ×     | ×       | ✓       | -              | ●      | ●       | ×        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | Fantastic Four   | ●                   | ✓   | ×        | ×     | ✓       | ✓       | -              | ●      | ●       | ✓        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | MPCLeague        | ●                   | ✓   | ×        | ×     | ✓       | ✓       | -              | ●      | ●       | -        | ✓          | ✓             | ✓    |                |            |                |                           |
|           | Tetrad           | ●                   | ✓   | ✓        | ×     | ✓       | ✓       | ✓              | ●      | ●       | ×        | ✓          | ✓             | ✓    |                |            |                |                           |

**Oblivious Transfer (OT):** OT is a protocol where a sender sends one of several possible pieces of information to a receiver without knowing which piece of information the receiver obtained.

**Limitation:** In OT protocols, the amount of data transmitted is proportional to the bit length of the input data. This implies a considerable amount of communication complexity. The communication overhead can cause significant WAN delays.

**Garbled Circuits (GC):** In 1986, Andrew Yao proposed GC [154] – a cryptographic protocol that enables two parties to jointly evaluate a function over their private inputs without the presence of a trusted party. Yao’s GC transforms any function into a securely-evaluated function by modeling it as a Boolean circuit. The inputs and outputs of each gate are masked so that the party executing the function cannot discern any information about the inputs or intermediate values of the function.

**Limitation:** The computation and communication overhead of ML execution utilizing the GC protocol is controlled by the number of neurons in each ML layer [126]. Each ML model layer contains thousands of values, which may result in large computing and communication overheads.

**Security Notions:** In SMPC, a set of mutually distrusting parties wish to jointly and securely compute a function of their inputs. This computation should be performed in such a way that each party obtain the proper result, and none of the parties learn anything beyond their prescribed output. A precise definition of security is required in order to prove that the SMPC protocol provides secure computation. A number of definitions have been proposed to ensure several security parameters that encompass most SMPC tasks. Here, we discuss this set of security parameters [93, 161].

- **Privacy** – No party should learn more than its prescribed output. More specifically, only what can be inferred from the output itself should be known about other parties’ inputs.
- **Correctness** – Each party is guaranteed that the output it receives is correct.
- **Independence of input** – The corrupt parties must choose their inputs independently of the honest parties’ inputs.
- **Robustness** – Regardless of the adversary’s actions, all parties can compute the protocol’s output. There are various levels of robustness:

- *Guaranteed Output Delivery (GOD)* – The strongest level is GOD, where honest parties are always certain to receive the output regardless of the adversary’s behaviors.
- *Fairness* – It is a weaker variant compared to GOD. It states that the adversary receives the output if and only if the honest parties receive the output.
- *Security with Selective Abort* – It is the weakest security notion, where the adversary can selectively deprive the honest parties from the output. In selective abort, at the end of computation, the adversary receives the output for certain values, but can prevent honest parties from receiving their outputs.

Using these security concepts, we later discuss the privacy and security guarantees offered by the current PPML protocols.

## B MACHINE LEARNING AND DATASETS

ML automates the analysis of datasets, producing models that reflect general relationships as found in the data. The two fundamental phases of ML are the training where the model is trained on input data, and the inference stage, where the trained model is put to use. ML techniques are divided into three classes, characterized by the nature of data available for analysis [37].

- **Supervised learning** – In this type, each input training example has a corresponding output that is also referred to as label. The objective is to train a model that can map the input examples to their outputs as accurately as possible. Examples of applications using supervised learning include: Image classification, text classification, spam filtering, machine translation, etc [137].
- **Unsupervised learning** – Uses ML algorithms to analyze and cluster unlabelled datasets. These algorithms discover hidden patterns in data without the need for human intervention. Examples of applications using unsupervised learning include: Recommendation system, anomaly detection, etc [79].
- **Reinforcement learning** – It is neither based on supervised nor unsupervised learning. The algorithm learns by exploring the environment and taking actions, thus maximizing cumulative rewards. It works with data in sequences of actions, observations, and rewards. Examples of reinforcement learning applications can be found in a plethora of areas such as self-driving cars, gaming and healthcare [150].

Following, we present a list of popular ML models that are trained in a privacy-preserving manner.

- **Linear Regression (LR)** – This ML approach models the relationship between two variables: a dependent variable  $y$  and an independent variable  $x$ . If a model only features an independent variable, then it is called a simple LR model; if it features more than one, then it is called multi-linear regression [131].
- **Convolutional Neural Network (CNN)** – It is a Deep Neural Network (DNN) often used in matrix-based applications like image recognition. As the name indicates, CNN uses convolution operations performed on the input data with a filter or kernel to produce feature maps [20].
- **Logistic Regression (LoR)** – A generalized regression model modeling the probability of a discrete outcome given an input variable. It makes use of a logistic function and can be used to describe certain nonlinear relations [80]. The most common LoR models produce a binary outcome, that is a result that can assume

two values (yes/no, true/false etc.). Multinomial LoR can be used to model scenarios with more than two discrete outcomes.

- **LeNet** [89] – It is a CNN architecture that was first proposed by LeCun *et al.* and was used in the automatic detection of zip codes and digit recognition. The network contains 2 convolutional and 2 fully connected layers. LeNet is the first architecture that shows the application of CNNs on a real-world dataset.
- **AlexNet** [86] – This network is the winner of the ImageNet ILSVRC-2012 competition [4]. It consists of 5 convolutional layers and 3 fully connected layers, uses a batch normalization layer (for stability and efficient training), and has about 60 million parameters. Achieved an impressive result of about 10.8% lower than the runner up for a top-5 error. This was accomplished by training the deep CNN using graphics processing units (GPUs). AlexNet made a big impact in computer vision and artificial intelligence, and the original paper has accumulated more than 110,000 citations according to google scholar at the time of writing this paper.
- **VGG16** [136] – Won the first place in the localisation task and second place in the classification task of the ILSVRC-2014 competition [5]. It consists of 16 layers and has about 130 million parameters. VGG16 shows we can build very DNNs, e.g. 16-19 weight layers, by utilizing very small convolution filters (size  $3 \times 3$ ). VGG16 shows that very DNNs can generalize well to various datasets and reproduce SotA results.

Data is an essential component of the ML model. Below, we discuss popular datasets used for training the PPML techniques:

- **MNIST** – A collection of hand-written digits. Consists of 60,000 images in the training set and 10,000 in the test set. Each image is a  $28 \times 28$  pixel image, the element in the matrix ranges from zero to 255 along with a label between 0 and 9 [88].
- **CIFAR** – It consists of 60,000 colored images out of which 50,000 are used for training and the remaining 10,000 are used for testing. These images are grouped into 10 mutually exclusive classes: airplanes, automobile, bird, cat, deer, dog, frog, horse, ship, truck [85].
- **Tiny ImageNet** – It is a subset of the ImageNet dataset in the ILSVRC [2]. Tiny ImageNet dataset consists of 100,000 training samples and 10,000 test samples with 200 different classes. Each sample is cropped to a size of  $64 \times 64 \times 3$  [87].
- **UCI** – The University of California, Irvine (UCI) ML repository [49] is a collection of more than 600 datasets of varying sizes used in ML tasks to evaluate the effectiveness of an ML algorithm. Some of the more well-known datasets in this repository are the Iris dataset [56] and YouTube Spam Collection [22].
- **iDASH** – iDASH Privacy & Security Workshop holds an annual secure genome analysis competition, which uses a customized genetic dataset every year to create PPML algorithms that classify sensitive genome data [141].
- **Yelp Dataset** – It contains 6,990,280 reviews of 150,346 businesses from the website Yelp, which publishes crowd-sourced reviews of various establishments [73].
- **AG News Classification Dataset** – The dataset is constructed from 4 of the largest classes in *AG’s corpus of news articles* [66]. The original corpus contains over a million news articles from more than 2,000 news sources and was made by the prior academic news search engine *ComeToMyHead*. The classification dataset contains 120,000 training samples and 7,600 testing samples [159].

- **IMDB Dataset of Movie Reviews** – Also known as the *Large Movie Review Dataset* [3, 98] is a dataset for binary sentiment classification containing an equal amount of movie review samples for training and testing (50,000 samples).
- **DBPedia Ontology Classification Dataset** – A crowd-sourced collection of the most commonly used infoboxes within Wikipedia [45]. Dataset consists of 14 ontology classes and each class has 40,000 training samples and 5,000 testing samples [158]. As such, the total size of the training dataset is 560,000, plus 70,000 testing [160].