Privacy-Preserving Machine Learning Using Functional Encryption: Opportunities and Challenges

Prajwal Panzade, *Member, IEEE*, Daniel Takabi, *Member, IEEE*, and Zhipeng Cai, *Fellow, IEEE*

***Abstract*—With the advent of functional encryption (FE), new possibilities for the computation of encrypted data have arisen. FE enables data owners to grant third-party access to perform specified computations without disclosing their inputs. It also provides computation results in plaintext, unlike fully homomorphic encryption (FHE). The ubiquitousness of machine learning (ML) has led to the collection of massive private data in the cloud computing environment. This raises potential privacy issues and underscores the need for more private and secure computing solutions. Numerous efforts have been made in privacy-preserving ML (PPML) to address security and privacy concerns. There are approaches based on FHE, secure multiparty computation (SMC), and, more recently, FE. Compared to FHE-based PPML techniques, FE-based PPML is still in its infancy. In this article, we provide a survey of PPML works based on FE, summarizing state-of-the-art literature. We focus on inner product-FE, function-hiding inner product encryption, and quadratic-FE-based ML models for PPML applications. We analyze the performance and usability of the available FE libraries and their applications to PPML. We also discuss future research directions for FE-based PPML approaches. To the best of our knowledge, this is the first work to survey FE-based PPML approaches.**

***Index Terms*—Computation on encrypted data, functional encryption (FE), privacy-preserving machine learning (PPML), trustworthy AI.**

I. Introduction

MACHINE learning (ML) techniques have become deeply integrated across domains like computer vision, natural language processing, and speech processing, enabling myriad applications. Increasingly, real-world ML relies on a cloud-based framework, epitomizing ML as a Service (MLaaS) [1]. Sectors with stringent regulations like banking, government, insurance, and healthcare are progressively migrating their data and ML services to the cloud. This shift highlights the escalating need for robust, secure computational solutions that safeguard data and model privacy in cloud-based ML. Consequently, research has focused on privacy-preserving ML (PPML) [2], [3], [4], targeting data and model privacy issues throughout ML stages.

Established techniques like fully homomorphic encryption (FHE) [5] and secure multiparty computation (SMC) [6] are fundamental PPML methods. Concurrently, functional encryption (FE) [7] is evolving. FHE allows computation on encrypted data without decryption. SMC enables joint computation while preserving individual privacy. In contrast, FE permits computation on encrypted data, yielding plaintext results.

Gilad-Bachrach et al. [8] proposed a pioneering method that involves the transformation of a pretrained neural network into a cryptographic model termed CryptoNet. This method enables secure transmission of homomorphically encrypted data from data owners to a central server, facilitating the reception of an encrypted inference. Similarly, Hesamifard et al. [9] introduced CryptoDL, a novel methodology utilizing FHE for privacy-preserving inference on pretrained convolutional neural networks (CNNs). In a distinct study, Graepel et al. [10] detailed a binary classification technique in their work, ML Confidential, which integrates polynomial approximations and FHE. Furthermore, Mohassel and Zhang [11] presented SecureML, elaborating on an efficient two-party protocol for training linear regression, logistic regression, and neural network models while ensuring data privacy. Wagh et al. [3] proposed a three-party computation protocol specifically designed for privacy-preserving training and inference in CNNs in their study, SecureNN.

In this study, we explore PPML methodologies leveraging FE. Within the MLaaS framework, the model resides on the server, and one or more clients are responsible for the training process. At times, the server may possess pretrained models. Typically, in FHE-based ML, models are trained on unencrypted data, with inferences derived from encrypted data. In this scenario, a client transmits encrypted data to the server, which conducts tasks such as classification using the pretrained ML model on the client data and delivers the prediction outcomes in the ciphertext. Notably, the server performs computations on encrypted data without learning the inputs, ensuring that only the data owner can access the actual result. Contrastingly, in FE, the server generates computation results in plaintext utilizing a specific key, enabling partial decryption required for computation (refer to Fig. 1). The subsequent

Manuscript received 11 October 2023; revised 15 November 2023; accepted 20 November 2023. Date of publication 1 December 2023; date of current version 21 February 2024. This work was supported in part by the National Science Foundation under Grant 2020636, Grant 2054968, Grant 2118083, Grant 2315596, and Grant 2244219; and in part by the Microsoft Faculty Fellowship Program. *(Corresponding author: Prajwal Panzade.)*

Prajwal Panzade and Zhipeng Cai are with the Department of Computer Science, Georgia State University, Atlanta, GA 30302 USA (e-mail: ppanzade1@student.gsu.edu; zcai@gsu.edu).

Daniel Takabi is with the School of Cybersecurity, Old Dominion University, Norfolk, VA 23529 USA (e-mail: takabi@odu.edu).

Digital Object Identifier 10.1109/JIOT.2023.3338220

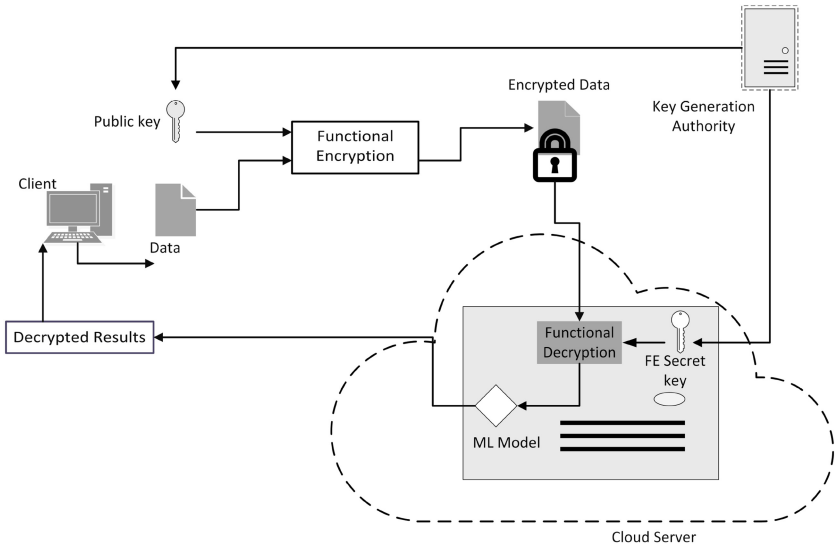


Fig. 1. Overview of PPML using FE.

procedures closely parallel those of FHE. Within the realm of PPML, the prospect of producing plaintext results over encrypted data without full decryption is notably intriguing for certain applications. It is worth noting that both FHE and FE-based PPML necessitate high computational costs.

This study thoroughly examines multiple research papers on FE-based PPML to provide a comprehensive overview to the research community. Focusing on FE-based PPML approaches amidst various existing methods, recent efforts have centered on developing FE-based systems, broadly categorized into inner product FE (IPFE)-based and quadratic FE (QFE)-based methodologies. Ligier et al. [12] proposed an approach for privacy-preserving classification on IPFE-encrypted data, and Xu et al. [13] introduced an IPFE-based deep neural network approach for image classification on the MNIST data set. Additionally, Panzade and Takabi presented methods for faster computation of secure activation functions using function-hiding inner product encryption (FHIPE) for PPML [14] and a privacy-preserving neural network training framework using IPFE and FHIPE [15]. Ryffel et al. [16] developed a system using QFE with adversarial training for privacy-preserving predictions. Marc et al. [17] presented fully fledged FE libraries and their applications in privacy-enhanced ML models.

The following are the primary contributions of this article.

1) We present a basic but substantial theoretical foundation to help researchers understand current approaches to FE-based PPML.
2) We provide a thorough review of the literature on FE-based PPML, emphasizing the strengths and shortcomings of the various approaches to assess how they supplement one another.
3) We examine the current constraints that prohibit the implementation of existing FE-based PPML solutions in real-world settings, mostly due to issues with efficiency and usability.
4) We provide research directions to intensify existing works in terms of time performance and security that the research community may pursue in the coming years.

II. Background Knowledge

FE is a generalization of public-key encryption that allows a key holder to compute a particular function of encrypted data using constrained secret keys [7]. Here, this function is called functionality. For example, an FE scheme may be particularly designed to compute inner products; in this case, the functionality becomes an inner product. In the FE scheme, a key management authority with a master secret key generates a secret key sk_{fe}; a decryptor can use that to compute a function on an encrypted message x. The symbols and acronyms used in this article are given in Table I. This section summarizes the two major FE schemes, IPFE and QFE, used by PPML approaches.

A. Inner Product Functional Encryption

The decisional Diffie–Hellman (DDH) assumption underpins the method outlined by Abdalla et al. [18]. Let GroupGenerator be a probabilistic polynomial-time (PPT) algorithm with input security parameter 1^λ, which produces a triplet $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a group of order p created by g *in* $\mathbb{G}$. The tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) are computationally indistinguishable, according to DDH, where $(\mathbb{G}, p, g) \leftarrow \text{GroupGenerator}(1^\lambda)$, and $a, b, c \in Z_p$ are chosen uniformly and independently at random.

The Π_{ipfe} = (Setup, Encrypt, KeyDerivation, Decrypt) FE scheme for IPFE in DDH is as follows.

Setup$(1^\lambda, 1^l)$*:* This algorithm samples $(\mathbb{G}, p, g) \leftarrow \text{GroupGenerator}(1^\lambda)$ and $s = (s_1, \ldots, s_l) \leftarrow Z_p^l$, sets $\text{mpk} = (h_i = g^{s_i})_{i\in[\ell]}$ and msk = s and finally returns a pair of (mpk, msk).

Encrypt(mpk, x): This algorithm takes mpk and message $x = (x_1, \ldots, x_l) \in Z_p^l$ as input, chooses random number $r \leftarrow Z_p$, computes $\text{Ct}_0 = g^r$ and, for each $i \in [l]$, $\text{Ct}_i = h_i{}^r \cdot g^{x_i}$ and returns ciphertext Ct.

KeyDerivation(msk, y): This algorithm takes msk and vector $y = (y_1, \ldots, y_l) \in Z_p$ as input and outputs key sk_{fe}.

TABLE I
SYMBOLS AND ACRONYMS USED IN THIS ARTICLE

Acronym / symbol	Description
mpk	Master Public Key
msk	Master Secret Key
sk_{fe}	FE Key for IPFE
sk_{qe}	FE key for QFE
Ct	Ciphertext
IND-CPA	security against chosen-plaintext attacks
ERT	Extremely Randomized Trees
IPFE	Inner-product Functional Encryption
QFE	Quadratic Functional Encryption
FHIPE	Function-Hiding Inner Product Encryption

Decrypt(mpk, Ct, sk_{fe}): This algorithm takes the master public key, ciphertext, and sk_{fe} for vector y as input and outputs the discrete logarithm in basis g of $\prod_{i\in[l]} Ct_i^{y_i}/Ct_0^{sk_{fe}}$.

Correctness: The method's correctness is demonstrated as follows [18]:

$\forall(mpk, msk) \leftarrow \mathrm{Setup}(1^\lambda, 1^l)$, all $y \in \mathbb{Z}_p^l$ and $x \in \mathbb{Z}_p^l$ for $\mathrm{sk}_{fe} \leftarrow \mathrm{KeyDerivation}(msk, y)$ &Ct $\leftarrow \mathrm{Encrypt}(mpk, x)$.

$$\begin{aligned}\mathrm{Decrypt}\big(mpk, \mathrm{Ct}, \mathrm{sk}_{fe}\big) &= \frac{\prod_{i\in[l]} \mathrm{Ct}_i^{y_i}}{\mathrm{Ct}_0^{\mathrm{sk}_{fe}}} \\ &= \frac{\prod_{i\in[l]} \big(g^{s_i r + x_i}\big)^{y_i}}{g^{r\left(\sum_{i\in[l]} y_i s_i\right)}} \\ &= g^{\sum_{i\in[l]} y_i s_i r + \sum_{i\in[l]} y_i x_i - r\left(\sum_{i\in[l]} y_i s_i\right)} \\ &= g^{\sum_{i\in[l]} y_i x_i} \\ &= g^{\langle x, y\rangle}.\end{aligned}$$

B. Function Hiding Inner Product Encryption

Function hiding inner product encryption [19] is another variant for computing inner products on encrypted data with three key differences. First, FHIPE provides the special feature of hiding the functionality. Second, it uses only one master key for key derivation and encryption. Third, it is more secure than IPFE as it provides simulation-based security in addition to indistinguishability-based security (IND-CPA). The details on security definitions are discussed in Section V-D. For more information on FHIPE, we refer the readers to [19].

C. Quadratic Functional Encryption

A QFE scheme uses bilinear groups (also known as pairing groups) and has been proposed by [20] and [21]. In the case of QFE-based PPML, we refer to schemes proposed by Ryffel et al. [16]. Here, GroupGenerator is a PPT algorithm on inputting 1^λ returns $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, e)$ of an asymmetric bilinear group, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order p (for a 2λ-bit prime p) and g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible pairing, i.e., it can be efficiently computable, nondegenerated, and bilinear: $e(g_1^\alpha, g_2^\beta) = e(g_1, g_2)^{\alpha\beta}$ for any scalars $\alpha, \beta \in \mathbb{Z}_p$. Therefore, $g_T \coloneqq e(g_1, g_2)$ which makes the group $\mathbb{G}_T$ of order p, where p is prime. For any $s \in \{1, 2, T\}, n \in \mathbb{N}$, and vector $u \coloneqq \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \in \mathbb{Z}_p^n$, it is denoted by $g_s^u \coloneqq \begin{pmatrix} g_s^{u_1} \\ \vdots \\ g_s^{u_n} \end{pmatrix} \in \mathbb{G}_s^n$.

Similarly, for any vectors $u \in \mathbb{Z}_p^n, v \in \mathbb{Z}_p^n$, It is denoted by $e(g_1^u, g_2^v) = \prod_{1-1} e(g_1, g_2)^{\mathbf{u}_t \cdot \vec{v}_1} = e(g_1, g_2)^{\mathbf{u}\cdot\boldsymbol{v}} \in \mathbb{G}_T$, since $\boldsymbol{u} \cdot \boldsymbol{v}$ denotes the inner product between the vectors $\boldsymbol{u}$ and v, that is, $\boldsymbol{u} \cdot \boldsymbol{v} \coloneqq \sum_{i=1}^n u_1 v_i$.

Ryffel et al. [16] built an efficient FE scheme shown below for the set of functions defined, for all $n, B_x, B_y, B_f \in \mathbb{N}^*$, as $\mathcal{F}_{n,B_x,B_y,B_f} = \{f : [-B_x, B_x]^n \times [-B_y, B_y]^n \to \mathbb{Z}\}$ where the functions $f \in \mathcal{F}_{n,B_x,B_y,B_f}$ are expressed as a set of bounded coefficients $\{f_{i,j} \in [-B_f, B_f]\}_{i,j\in[n]}$, and for all vectors $\boldsymbol{x} \in [-B_x, B_x]^n, \boldsymbol{y} \in [-B_y, B_y]$

$$f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i,j\in[n]} f_{i,j} x_i y_j.$$

The FE scheme is explained as follows.

Setup($1^\lambda, \mathcal{F}_{n,B_x,B_y,B_f}$):

$\mathcal{PG} \coloneqq (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, e) \leftarrow \mathrm{GroupGenenerator}(1^\lambda)$, s, t $\leftarrow \mathbb{Z}_p^n$, **msk** $\coloneqq$ (s, t),

mpk $\coloneqq (\mathcal{PG}, g_1^s, g_2^t)$

Return (mpk, msk).

Encrypt(mpk,(x, y)):

$\gamma \leftarrow \mathbb{Z}_p$, $\mathbf{W} \leftarrow \mathrm{GL}_2$, for all $i \in [n]$,

$\boldsymbol{a}_i \coloneqq (\mathbf{W}^{-1})^\top \begin{pmatrix} x_i \\ \gamma s_i \end{pmatrix}, \boldsymbol{b}_i \coloneqq \mathbf{W} \begin{pmatrix} y_i \\ -t_i \end{pmatrix}$

Return $\mathrm{Ct} \coloneqq (g_1^\gamma, \{g_1^{a_i}, g_2^{b_i}\}_{i\in[n]}) \in \mathbb{G}_1 \times (\mathbb{G}_1^2 \times \mathbb{G}_2^2)^n$

KeyDerivation(msk, f):

Return $\mathrm{sk}_{qe} \coloneqq (g_2^{f(s,t)}, f) \in \mathbb{G}_2 \times \mathcal{F}_{n,B_x,B_y,B_f}$

Decrypt(mpk, Ct $\coloneqq (\mathrm{g}_1^\gamma, \{g_1^{a_i}, g_2^{b_i}\})_{i\in[n]}$),

$\mathrm{sk}_{qe} \coloneqq (g_2^{f(s,t)}, f)$):

out $\coloneqq e(g_1^\gamma, g_2^{f(s,t)}) \cdot \prod_{i,j\in[n]} e(g_1^{a_i}, g_2^{b_i})^{f_{i,j}}$

Return log(out) $\in \mathbb{Z}$

Correctness:

For all $i, j \in [n]$

$$e\left(g_1^{d_i}, g_2^{b_j}\right) = g_T^{d_i \cdot b_j} = g_T^{x_i y_j - \gamma s_i t_j}$$

since

$$\begin{aligned}\vec{a}_i \cdot \vec{b}_j &= \left(\left(\mathbf{W}^{-1}\right)^\top \begin{pmatrix} x_i \\ \gamma s_i \end{pmatrix}\right)^\top \cdot \left(\mathbf{W}\begin{pmatrix} y_j \\ -t_j \end{pmatrix}\right) \\ &= \begin{pmatrix} x_i \\ \gamma s_i \end{pmatrix}^\top \mathbf{W}^{-1}\mathbf{W}\begin{pmatrix} y_j \\ -t_j \end{pmatrix} \\ &= x_i y_j - \gamma s_i t_j.\end{aligned}$$

Therefore,

out $= e(g_1^\gamma, g_2^{q(\vec{s},\vec{t})}) \cdot \prod_{i,j} e(g_1^{\vec{a}_i}, g_2^{\vec{b}_i})^{q_{i,j}} = g_T^{\gamma q(\vec{s},\vec{t})} \cdot g_T^{\sum_{i,j} q_{i,j} x_i y_j - \gamma q_{i,j} s_i t_j} = g_T^{\gamma q(\vec{s},\vec{t})} \cdot g_T^{q(\vec{x},\vec{y}) - \gamma q(\vec{s},\vec{t})} = g_T^{q(\vec{x},\vec{y})}$

We refer the readers to [16], [18], and [22] for more cryptographic details on IPFE and QFE schemes, respectively.

D. Neural Networks

The artificial neural network, often known as a neural network, is an ML model that is hierarchical and nonlinear, with several layers and several neurons in each layer. Each

layer of a neural network processes the input provided by the previous layer before passing it on to the next.

1) *Input Layer:* The preprocessed raw data or features extracted from raw data in a particular format make up the first layer of the neural network.
2) *Hidden Layer:* A neural network can have one or more hidden layers. The first hidden layer's neurons are linked to the input layer and followed by an activation function. Further hidden layers are fed with the previous layer's output. Weight values are associated with layers, and they are updated during the forward and backpropagation processes until convergence.
3) *Activation Function:* The activation function of a neuron in a neural network determines the output of that neuron given a single or group of inputs. In ML, there are several activation functions, such as sigmoid, rectified linear unit (ReLU), and tanh. The ReLU activation function is an example of one of the most frequently used activation functions. If the input value is less than zero, the ReLU activation function returns zero, and if it is greater than zero, it returns the same input.
4) *Output Layer:* The output layer of a neural network is the final layer of neurons that provides the network's output.

E. Polynomial Neural Network

Polynomial neural networks are the kinds of neural networks that primarily facilitate linear components like fully connected layers, convolutions with average pooling, and activation functions approximated using polynomials. They have demonstrated fairly high accuracy for the relatively simple benchmarks in image recognition tasks [8], [23]. They have also been used to propose the applicability of novel ML protocols in various early-stage implementations, as presented in [24] and [25]. The simplicity of the operations upon which polynomial neural networks are built ensures high efficiency, particularly for gradient computations. Research works, such as [26], have demonstrated that they can reach convergence rates comparable to those of networks with nonlinear activation functions.

III. Scope

Our analysis of the FE-based PPML methodologies is divided into two parts. To begin, we conduct a thorough review of existing methodologies and emphasize their salient features and functions. Second, we evaluate these tools in practice by comparing their usability, complexity, and performance across various case study scenarios. The secure computation ecosystem encompasses a wide variety of tools. On the low level, there exist math libraries that facilitate the construction of FE implementations, for example, by efficiently implementing algorithms useful for generic lattice-based cryptography. Then there are FE libraries that implement certain schemes and provide slightly more advanced application programming interfaces (APIs), such as setup, encrypt, key generation, and decrypt. These libraries abstract away computation features like parameter selection, encryption, and decryption by providing a higher level language in which developers can implement their computation.

The primary objective of this work is to comprehend the landscape of MLaaS in data-sensitive scenarios via PPML computing, for example, when data supplied to third parties for processing is encrypted. PPML ensures the privacy and confidentiality of input data. Additionally, they alleviate excessive pressures on the client endpoint in computing as the cloud server does most of the computing part. Finally, they may be used in ML situations where clients can contribute data toward a training or inference goal. PPML is frequently used in conjunction with other approaches. We discuss the application of all PPML approaches based on FE just at these crossing places. Although FHE is a widely used technique in PPML, we only include FE-based approaches because of fewer available articles to the research community on this topic.

The second problem is associated with the techniques' deployability. Numerous frameworks and technologies make advanced ML accessible to data scientists who are not necessarily professionals in computer sciences. However, adopting these frameworks for use with PPML is challenging. We consider usability enhancements in this area if the proposal fits one of the following two requirements. To start, it makes the solution more adaptable to existing ML frameworks (e.g., by giving tools that reduce total programming effort). As such, our study incorporates works that propose APIs, compilers, or other significant practical tools that aid in the implementation and deployment of theoretical ideas into real applications, hence increasing their usability. Second, when an open-source implementation supplements the concept. Apart from facilitating future revisions to the approach, open-source implementations ensure that the results are reproducible. Additionally, we verify whether the work contains references to open-source implementations or is released independently (e.g., by visiting the authors' websites or GitHub repositories).

In summary, we examine the following for each of the proposals utilizing FE-based PPML: 1) the problem being addressed (training, inference, or both); 2) the ML model used; 3) the specific FE techniques involved (IPFE or QFE); and 4) the efficiency and usability considerations examined.

IV. Overview of FE-Based PPML Models and Libraries

A. What Has Been Done?

There are two variations of the FE-based PPML methodologies available in the literature. The first one uses IPFE, whereas the other one uses QFE. The IPFE-based methodologies involve training and inference, whereas the QFE methodologies involve simply the inference stage of ML.

1) Inner Product FE-Based Machine Learning: In this type of methodology, the inputs are encrypted using IPFE. Then during the activation, inner products between encrypted inputs and weight matrices are unfolded based on the special property of FE. Later, the neural network operations are done similarly to regular neural networks. Here, both forward propagation and backpropagation can be made secure using

FE. This methodology supports both training and prediction over encrypted data, unlike the QFE-based approaches.

2) Quadratic FE-Based Machine Learning: In QFE-based methodologies, the training phase happens similarly to the regular neural networks. It is also worthwhile to note that they train the neural networks with plain data. In the prediction phase, encrypted data are fed to the neural network. It undergoes the process of polynomial approximation, and then the other steps are applied similar to regular neural networks. These approaches are found to be faster than IPFE-based approaches.

B. Cryptography Libraries for Functional Encryption Implementation

Presently, there are only two dedicated libraries that focus on implementing state-of-the-art FE schemes. The first one is called CiFEr, and the other is GoFE. Three entities are involved in FE and decryption: 1) an encryptor; 2) a decryptor; and 3) a key management authority. An encryptor encrypts the data and obtains ciphertext. The decryptor decrypts the ciphertext received from the encryptor. The key management authority handles the responsibility of generating a variety of cryptographic keys. In the FE scheme, based on the involvement of encryptors, it can be either single-input or multi-input. We detailed both of these libraries and supporting cryptography libraries used in the FE-based PPML works as follows.

1) CiFEr: CiFEr[1] [17] is an FE library developed by the FENTEC group that is written in the C language. This library allows developers to use functions to perform various FE operations like encrypt, decrypt, and key generation. This library is built in such a way that the predefined functions can be directly called without setting many parameters. Here, the cryptographic key generation is abstract to the user, and users set the security parameters in terms of bits. It provides both single-input and multi-input FE implementations commonly observed in FE-based PPML approaches.

2) GoFE: GoFE[2] [17] is another library provided by the FENTEC group that is written in Golang. Like CiFEr, it also provides an option to use FE functions. Both CiFEr and GoFE have the same set of state-of-the-art FE implementations. Their performance is based on the underlying programming languages.

3) FLINT: FLINT[3] [27] is a cryptography library that is used for performing number theory-related operations. It is written in *C*. Unlike GoFE and CiFEr, it does not dedicatedly provide FE functionality. In some of the FE-based works, this library is used for the implementation of FE schemes.

4) Charm: Charm[4] [28] is a framework designed for implementing various advanced cryptosystems. It is built using Python to decrease development time and code complexity while fostering component reuse.

5) PBC: Pairing-based cryptography (PBC)[5] [29] is a *C* library that enables rapid development of cryptosystems based on pairings. It implements a bilinear cyclic group abstractly, hiding the programmer from mathematical details. Both PBC and Charm are used in some of the FE-based PPML implementations.

[1]https://github.com/fentec-project/CiFEr
[2]https://github.com/fentec-project/gofe
[3]https://flintlib.org/
[4]https://github.com/JHUISI/charm
[5]https://crypto.stanford.edu/pbc/

V. Insights

Recall that we consider the FE-based PPML works focused on training and/or inference over encrypted data. The comparison details of these works in terms of functionality and performance is given in Tables II and III, respectively.

A. Threat Model

Assumption: In FE-based PPML, it is assumed that there is a trusted independent key management authority that is responsible for the generation of the required keys. Such assumptions are common in all FE schemes [7], [18], [22], [32]. FE-based PPML techniques adhere to an honest but curious security model, in which both parties comply with the protocol while attempting to gain as much information as possible. In this case, the server follows the protocol but tries to learn additional information. Typically, the approach includes three primary components: 1) a key management authority; 2) a server; and 3) a client. The key management authority generates encryption and decryption keys. FE requires three distinct keys, namely, master public key (mpk), master secret key (msk), and FE key(sk_{fe}), that operate slightly differently than public-key cryptography techniques as discussed in Section II.

The research in this field assumes an MLaaS setting. In the case of training, the client provides the encrypted data, and the cloud server performs training using the provided encrypted data and obtains a trained model. In the case of inference, the server holds the trained model, and the client provides the encrypted data to the server and receives the prediction results in plaintext.

The client first encrypts the data using the master public key before sending it to the server. In some scenarios, the client may be required to preprocess the data before encryption; this varies by application. For example, computer vision applications may require scaling or transforming images prior to encryption. Normalization and standardization may be necessary in the case of structured data.

The server holds the model. The model is developed by training the neural network using client-supplied data. Because the server cannot see the data sent by the client, it must obtain the FE secret key sk_{fe} in order to execute functionality-based computations on the data. Either polynomial approximation uses such computations for QFE-based methods or inner product computation in IPFE-based methods discussed in the later sections.

B. IPFE-Based PPML

Fig. 2 depicts IPFE-based PPML. Out of the available works in this field dedicated to FE-based PPML [12], [13], and [15] use IPFE schemes.

TABLE II
COMPARISON OF FE-BASED PPML MODELS IN TERMS OF FUNCTIONALITY

Research works	FE type	Training	Prediction	ML model	Security
Ligier et al. [12]	IPFE	○	●	ERT	Selective IND-CPA
Xu et al. [13]	IPFE	●	●	5 layer NN	Selective IND-CPA
Sans et al. [30]	QFE	○	●	2 layer NN	Adaptive IND-CPA
Ryffel et al. [16]	QFE	○	●	2 layer NN	Adaptive IND-CPA
Carpov et al. [31]	IPFE and QFE	○	●	2 layer NN	Adaptive IND-CPA
Panzade et al. [15]	IPFE and FHIPE	●	●	5 layer NN	Selective IND-CPA and Simulation-based

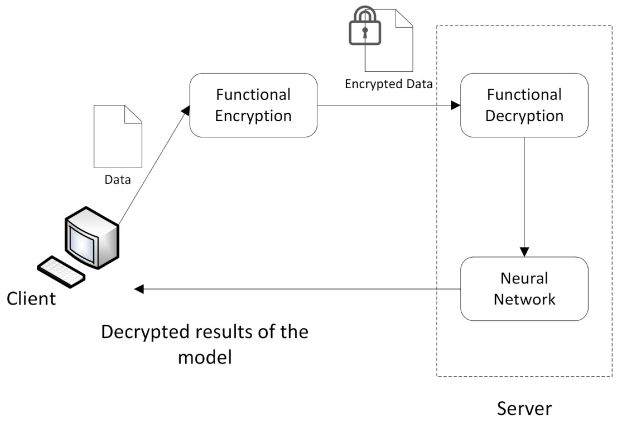


Fig. 2. IPFE-based PPML.

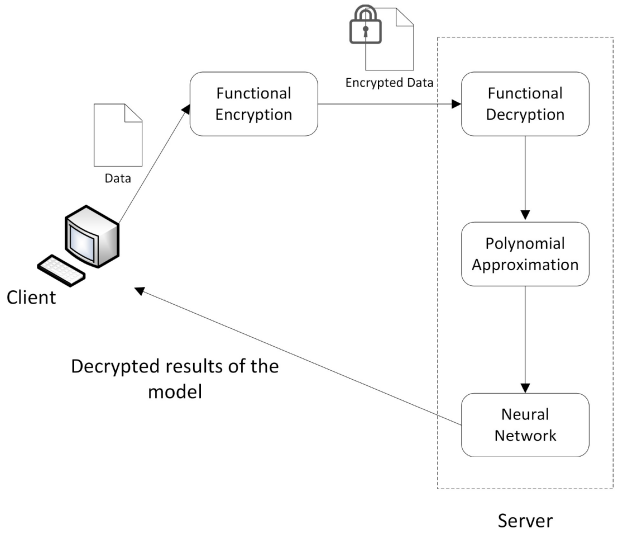


Fig. 3. QFE-based PPML.

1) Functional Encryption Scheme: The methodology presented in [12] is founded on the FE scheme proposed by [32]. In contrast, [13] adopts the FE scheme introduced by [18]. Moreover, [15] leverages FE schemes derived from both [18] and [19].

2) Training Phase: In the approach proposed by Ligier et al. [12], Extremely Randomized Trees serve as the ML model, functioning as an ensemble learning model based on decision trees. Their training process involves plain data, akin to regular ML models. For inference, the client obtains encryption keys and then encrypts their input data using IPFE and sends the ciphertext to the server. The server then decrypts it with FE secret keys to compute inner products and performs classification, without having access to inputs in plain.

The methodologies proposed in [13] and [15] employ a 5-layer neural network. In their IPFE-based approach, client inputs are initially encrypted before transmission to the server. The server, equipped with the FE key, executes the IPFE decrypt function to compute activation results in the first hidden layer.

The computation between the input vector and weight matrix in the first hidden layer is expressed as

$$A = \mathrm{ReLU}\big(sk_{fe}(W) \cdot \mathrm{Encrypt}(X) + b\big).$$

The output from this first hidden layer cascades through subsequent layers, indicating the feasibility of these methodologies functioning entirely on FE-encrypted data.

Training using encrypted data, as proposed by [13], demands 57 h for the 5-layer neural network model, whereas the same nonencrypted neural network under identical settings takes only 4 h. This significant difference underscores the computational gap between encrypted and nonencrypted data within neural networks.

In FENet proposed by Panzade and Takabi [15], alongside IPFE, the FHIPE scheme is employed. Notably, the FHIPE scheme enhances inner product computation while maintaining functionality secrecy, as discussed in Section II-B. Their work achieves over a 2× speedup compared to the approach in CryptoNN [13], while preserving comparable accuracy.

3) Inference Phase: The methodology proposed in [12] demonstrates the ability to make predictions on encrypted data in less than 0.1 s, achieving a validation accuracy of 95.64%. Similarly, [13] conducts inference on encrypted data by utilizing IPFE decryption functions, achieving a validation accuracy of 95.49%. On the other hand, FENet [15] achieves approximately 95% accuracy, with each inference taking less than 3 s to perform on FE-encrypted data.

C. QFE-Based PPML

Fig. 3 depicts QFE-based PPML. Three works use QFE schemes, which are defined and shown utilization toward ML problems in [16], [30], and [31].

1) Functional Encryption Scheme: This type of research works either uses the similar or modified approach proposed by Baltico et al. [22]. Works proposed by [16] and [30] are related to each other in terms of the FE scheme and neural

networks used. They proposed a novel FE scheme that can be used on multivariate quadratic polynomials.

2) Training Phase: The training phase in this type of methodology is done on plain data similar to regular neural networks [30] and [31]. In the work proposed by [16], the adversarial training approach is used in order to avoid information leakage after functional decryption. They optimized both the primary classification objective and the opposite of the collateral objective of a particular simulated adversary simultaneously using adversarial training.

3) Inference Phase: In the inference phase of all the works proposed in [16], [30], and [31], polynomial neural networks are used. Here, they use two-layer neural networks. Wherever QFE is used, a special type of neural network called the polynomial neural network is used discussed in Section II-E. The approaches based on QFE have not yet been used for training the model over encrypted data because of the complexities involved in the cryptographical aspects. Also, the present FE is defined for the usage of degree 2 polynomials. The methodology proposed by [16] and [30] requires less than 3 s to produce a prediction result. Moreover, in the case of QFE-based works, accuracy is shown a little higher. References [16] and [30] obtain an accuracy of 97.54% and 97.7%, respectively.

D. Security

1) Selective Security: Selective security is a notion of security for FE where the adversary commits to the challenge messages before seeing the scheme's public parameters. This allows for more efficient constructions as security only needs to hold for the predetermined challenge messages rather than all messages [18].

2) Adaptive Security: Adaptive security is a strong notion of security for FE where the adversary can query secret keys and choose challenge messages adaptively after seeing the scheme's public parameters. This requires security to hold for all possible messages, posing a greater challenge for achieving efficient constructions compared to selective security [16], [22].

3) Simulation-Based Security: Simulation-based security is a concept that strengthens the notion of security by focusing on the ability to simulate the actions of an adversary given only limited access to specific information. It ensures that the behavior of the encryption scheme in the real world is indistinguishable from an idealized scenario [19].

Theorem 1: Under the DDH assumption, the IPFE scheme given in Section II-A is *selectively secure against chosen-plaintext attacks* (IND-IPFE-CPA).

Theorem 2: Under the matrix DDH (MDDH) assumption, the QFE scheme given in Section II-C is *adaptively secure against chosen-plaintext attacks* (IND-QFE-CPA).

Theorem 3: An FHIPE scheme Π_{fhipe} is SIM-secure if, for all the effective adversaries $\mathcal{A}$, there exists simulator $\mathcal{S}$ in which the cases given in [19] are computationally indistinguishable.

We refer the readers to [16], [18], [19], and [22] for the security proofs of the theorems.

The security of the FE-based PPML approaches is dependent on the cryptographic security of the underlying FE schemes. As mentioned earlier, they offer three types of security: 1) adaptive security; 2) selective security; and 3) simulation-based security against chosen-plaintext attacks that can be found in more detail in [18], [19], and [22]. The approaches proposed in [12], [13], and [15] are selectively secure whereas the approaches proposed in [16], [17], and [30] show adaptive security based on the underlying crypto schemes. In addition to selective security, FENet [15] also offers sim-security.

E. Information Leakage and Attacks

Although FE schemes provide cryptographic security to the ML models, there is a risk of information leakage. The outputs obtained in the IPFE, FHIPE, and QFE-based models are in plaintext after the FE decryptions are done. So, if the server tries to retrieve part of the information from the obtained plain data after FE computations, there is a possibility of information leakage. Carpov et al. [31] discussed the type of information leakage. Also, an adversarial training-based approach proposed in [16] can avoid information leakage to some extent. We believe there is a scope of attacks like model inversion attacks [33] and direct inference attacks that ML security researchers can study. In order to make the FE models deployable, this potential threat should be taken into consideration.

F. Implementation Details

Almost all of the methodologies use the MNIST data set [34] for the experiments. MNIST contains 70 000 grayscale images of size 28×28 pixels of handwritten single digits from 0 to 9. Out of 70K images, the training set consists of 60K images, and the test set consists of 10K images. In [31], Census Income Data set [35] is used in addition to the MNIST data set. Census Income data was extracted from the 1994 Census Bureau database. It is a multivariate data set with 48 842 instances. There are 14 attributes like age, work class, and education with categorical and integer data. The task here is to predict whether a person's income exceeds \$50K/yr.

FE Libraries: Ligier et al. [12] implemented the IPFE scheme using FLINT library [27] whereas Xu et al. [13] used Charm library [28] for implementation of the IPFE scheme. Panzade and Takabi [15] used the CiFEr library [17] for FHIPE and IPFE implementation. Sans et al. [30] and Ryffel et al. [16] used PBC [29] and [28] library for implementation of QFE scheme. We have already detailed these libraries in Section IV-B.

Python Libraries: Ligier et al. [12] used sklearn [36] in python for implementation of ML model. Methodologies presented in [16] and [30] implemented the ML model in Tensorflow [37] in Python. Approaches proposed in [13] and [15] used just a Numpy [38] for the implementation of the ML model whereas [31] used Keras [39] in python for the implementation of ML model.

System Specifications: Table IV summarizes the machine specifications used for performing experiments by the various

TABLE III
COMPARISON OF FE-BASED PPML MODELS IN TERMS OF PERFORMANCE

Research works	Year	Training	Prediction	FE key generation	Encryption	Decryption	Accuracy
Ligier et al. [12]	2017	NA	$\leq$ 0.1s	12 ms	150 ms	69 s	95.64%
Xu et al. [13]	2018	57 hrs	not specified	not specified	not specified	not specified	95.49%
Sans et al. [30]	2019	NA	a few seconds	8 ms	8.1s	3.3 s	97.54%
Ryffel et al. [16]	2019	NA	$<$ 3 s	94 $\pm$ 5 ms	12.1 $\pm$ 0.3s	24 $\pm$ 9ms	97.7%
Carpov et al. [31]	2020	NA	not specified	not specified	not specified	not specified	90 %
Panzade et al. [15]	2023	26 hrs	$<$ 3 s	not specified	not specified	not specified	$\approx$ 95%

TABLE IV
IMPLEMENTATION DETAILS

Research works	Dataset	FE Libraries	Python Libraries	System specifications
Ligier et al. [12]	MNIST	FLINT	sklearn	Intel Core i7-4650U, 8GB RAM
Xu et al. [13]	MNIST	Charm	Numpy	Intel Core i7, 8GB RAM
Sans et al. [30]	MNIST	PBC and Charm	Tensorflow	Intel Core i5-6440HQ, 8GB RAM
Ryffel et al. [16]	MNIST	PBC and Charm	Tensorflow	Intel Core i7, 16GB RAM
Carpov et al. [31]	MNIST, Census Income	not specified	Keras	not specified
Panzade et al. [15]	MNIST	CiFEr	Numpy	Intel Xeon Gold 6230 R, 755 GB RAM

authors. The experiments done by all the research works are on Intel CPUs. As CPUs are involved, there is an issue of slow computation involved during training and prediction. This presents a notable contrast to the faster computational performance demonstrated by today's GPU-based ML models.

VI. DISCUSSION

A. Tradeoffs in FE-Based PPML

To ensure a fully PPML system, both training and prediction phases over encrypted data should be integrated into the system. Our studies observe that for a perfect PPML system, both training and prediction should be undergone over encrypted data. By doing so, the system ensures the privacy of the user's data for both tasks. This interesting concept of training and prediction without having access to plain data helps us develop a fully PPML system. Presently, only [13] and [15] focus on both of these tasks, but it is not efficient in terms of time.

Considering information leakage and adaptive security simultaneously is highly important. Security of the FE-based PPML schemes is entirely dependent on the underlying FE scheme. IPFE provides selective security, whereas QFE provides selective as well as adaptive security against chosen-plaintext attacks. In addition to selective security, FHIPE also provides simulation-based security. There is a theoretical advancement by [40] toward making IPFE adaptively secured, but FE-based PPML works do not yet implement it. Also, information leakage proposed by [31] is the only work that discusses this concept. This work also follows the traditional selective IPFE and adaptive QFE schemes. This raises a high need to consider both information leakage and adaptive security while building an FE-based PPML scheme.

There is no fit-for-all methodology. None of the existing solutions considers all the criteria for a perfect PPML system. However, we can rank the methodologies from each group, i.e., one from the IPFE-based scheme and another from the QFE-based scheme. In the IPFE-based scheme, [15] satisfies most of the criteria, whereas [16] performs the best in QFE-based methodologies. If prediction accuracy and privacy are considered, then [16] leads, whereas the overall performance of the system in terms of security, privacy, and accuracy is concerned, then [15] leads over others. Also, they provide more secure implementation based on FHIPE.

FE Versus FHE: Both FHE and FE cryptosystems are based on post-quantum lattice-based cryptography. Apart from the key distinction, i.e., the additional key requirement of FHE, FE has some pros and cons. The FE can be the best choice when the computation results on the encrypted data are expected to be plain. Although this is an advantage, present FE schemes support only inner product and quadratic computation on encrypted data. In this regard, FHE wins over FE as it gives computation capability for more complex polynomials. Both cryptosystems have the overhead of time and space complexity required for the encryption, decryption, and key generation processes. FHE is more developed in engineering aspects as leading companies like Microsoft [41] and IBM [42] have been working on it. FE has not yet received attention and lacks engineering aspects. Due to this, even though FE and FHE have a set of complex operations, FHE uses available modern hardware accelerator devices and engineering solutions and wins over FE.

B. Challenges and Future Research Directions

1) Functionality Enhancement in the FE Scheme: The current research done on FE is limited to two functionalities on integers: 1) inner product and 2) bilinear maps. Recall that functionality is a function computed over encrypted data to obtain decrypted results. As discussed, inner product functionality is used in the IPFE scheme, whereas bilinear maps are used in the QFE scheme. Because of this limited availability of functionalities, FE-based PPML methodologies still require enhancement. For example, QFE-based ML methods can use only 2-degree polynomial networks. Also,

FE cannot perform min/max and comparison operations. If enhancement in FE functionality is done, it will be helpful for PPML researchers to come up with practical solutions for supervised and unsupervised ML problems. However, these systems are not at a level to be used in real-world applications.

2) Improving Efficiency: Though FE-based PPML solutions have shown promising results in partially encrypted ML, there is a need to improve efficiency. Ciphertexts and keys generated with larger security parameters are slower compared to other cryptographic approaches. This becomes a threat when computations are performed on large data sets. Therefore, this efficiency issue stands as a big challenge in FE-based PPML methodologies. Moreover, this can appear as a big problem when FE-based PPML is considered in the case of resource-constrained Internet of Things (IoT) devices. To solve this problem, there is a need for lightweight FE schemes that can be run with limited resources, eventually improving efficiency.

3) Improvement in Security and Privacy of FE Scheme: As discussed in the previous section, the FE-based PPML methodologies' security depends on the underlying FE scheme. IPFE schemes [18], [32] are selectively secured against chosen-plaintext attacks under the DDH assumption. QFE schemes [16], [22] are adaptively secured against chosen plaintext attacks under MDDH assumption. These FE schemes may lag in stricter security applications such as the defense domain. Information leakage in FE-based PPML schemes needs to be explored and improved. In addition to this, ML attacks are also interesting to consider in this area.

4) Enhancing Privacy-Preserving Neural Networks: Today's FE-based works have shown their demonstration of only up to 5-layer neural networks. There is a scope to enhance the structure of neural networks. As discussed above, QFE-based methods are limited to using only degree-2 polynomial networks. So, the enhancement of these networks is somehow dependent on the underlying FE scheme. Implementation of various activation functions is also dependent on the underlying FE scheme. Apart from this, FE is not at a level to implement complex deep learning CNNs like VGGNet, AlexNet, and GoogleNet. Methods like transfer learning can be applied if there exists a functionality that can decrypt the encrypted model and retrieve the saved parameters like general CNNs.

5) Training-Centric PPML Systems: As the PPML is growing day by day, training over encrypted data is gaining the attention of researchers [3]. Presently, FE-based PPML schemes proposed in the literature are more inference-centric. Although the work proposed in [13] and [15] has successfully proposed a way for training a neural network using FE, they lack computational efficiency. There is a high need for PPML systems that can perform both training and inference over encrypted data.

6) Training From Multiple Data Sources: As far as ML is concerned, there are methodologies that focus on training models from multiple data sources in [3]. The enhanced versions of the FE scheme, like multi-input FE [43], [44], can be used to serve this purpose. These schemes facilitate using multiple vectors to perform computation based on inner product functionality. Such schemes could leverage the problem of training the model possessed by a cloud server for training from multiple data sources.

7) Using Multiauthority and Decentralized Extensions of FE: The FE scheme used in PPML services requires a trusted third party or trusted authority to be involved in generating the keys. The scheme proposed by [45] can be used to avoid the involvement of trusted authorities. In addition to this, there is a great scope for making the FE schemes used in PPML decentralized by using the approaches proposed by [46] and [47].

8) Need for Open-Source Library Support: FE has gained widespread attention from researchers as far as the theoretical aspects are concerned. However, it still lacks practical library implementations compared to its predecessor, FHE. For example, HElib [42] by the IBM research group, and SEAL [41] by the Microsoft research group are great tools for performing the FHE-related task. Presently, commendable efforts have been taken by the FENTEC group [17] for providing *C* and Go language versions of FE libraries, namely, CiFEr and GoFE. These libraries are in the development stage and are not yet ready to be deployed in production. So, there is a considerable need for practically applicable FE libraries.

9) Need for Hardware Acceleration Support: In all the implementations of FE-based PPML methodologies that have been studied in this article, computations are carried out on the CPU. Additionally, there has been no GPU support available for FE implementation. Due to these limitations, we lack the opportunity to leverage the highest GPU computing power available today. Real-world ML applications involve training on massive data sets. Therefore, performing computations in such applications is highly time consuming when done on the CPU. The work proposed by [48] uses System-on-a-Chip (SoC) implementation to provide acceleration support but requires special hardware. If easy-to-use hardware acceleration support using widely available GPUs or FPGAs were integrated into such applications, it would significantly expedite computations. This acceleration would eventually aid in making FE deployable in real-world applications.

10) Improving Scalability and Application to Nonimage Data Sets: Present works focused on FE-based PPML are limited to using MNIST-like small data sets for their experiments, and there are no implementations for larger data sets. In addition to this, the applications to nonimage data sets like text corpus may also be explored. Therefore, this stands as a future research direction in improving FE utilization in real-world applications.

VII. Conclusion

PPML has gained widespread attention among industry and academic researchers in recent years. Although approaches based on FHE and SMC have been extensively studied, FE-based solutions are still less investigated. In this article, we provide a summary and survey of PPML approaches based on FE. Our analysis assessed the extent to which previous work has addressed the PPML using FE and identified key

weaknesses in this area. Additionally, our analysis demonstrates that FE-based approaches could significantly contribute to the PPML issues, but there exist challenges that should be addressed to achieve practical solutions. We hope that this effort paves the path for the research community to investigate this emerging yet important topic.

REFERENCES

[1] M. Ribeiro, K. Grolinger, and M. A. Capretz, "MLaaS: Machine learning as a service," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2015, pp. 896–902.

[2] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," in *Proc. Privacy Enhanc. Technol.*, 2018, pp. 123–142.

[3] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-party secure computation for neural network training," in *Proc. Privacy Enhanc. Technol.*, 2019, pp. 26–49.

[4] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable provably-secure deep learning," in *Proc. 55th Annu. Design Autom. Conf.*, 2018, pp. 1–6.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[6] P. Bogetoft et al., "Secure multiparty computation goes live," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2009, pp. 325–343.

[7] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Proc. Theory Cryptogr. Conf.*, 2011, pp. 253–273.

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

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

[10] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Proc. Int. Conf. Inf. Security Cryptol.*, 2012, pp. 1–21.

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

[12] D. Ligier, S. Carpov, C. Fontaine, and R. Sirdey, "Privacy preserving data classification using inner-product functional encryption," in *Proc. ICISSP*, 2017, pp. 423–430.

[13] R. Xu, J. B. Joshi, and C. Li, "CryptoNN: Training neural networks over encrypted data," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2019, pp. 1199–1209.

[14] P. Panzade and D. Takabi, "Towards faster functional encryption for privacy-preserving machine learning," in *Proc. 3rd IEEE Int. Conf. Trust, Privacy Security Intell. Syst. Appl. (TPS-ISA)*, 2021, pp. 21–30.

[15] P. Panzade and D. Takabi, "FENet: Privacy-preserving neural network training with functional encryption," in *Proc. 9th ACM Int. Workshop Security Privacy Anal.*, 2023, pp. 33–43.

[16] T. Ryffel, E. Dufour-Sans, R. Gay, F. Bach, and D. Pointcheval, "Partially encrypted machine learning using functional encryption," 2019, *arXiv:1905.10214*.

[17] T. Marc, M. Stopar, J. Hartman, M. Bizjak, and J. Modic, "Privacy-enhanced machine learning with functional encryption," in *Proc. Eur. Symp. Res. Comput. Security*, 2019, pp. 3–21.

[18] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2015, pp. 733–751.

[19] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Proc. Int. Conf. Security Cryptogr. Netw.*, 2018, pp. 544–562.

[20] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[21] A. Joux, "A one round protocol for tripartite Diffie–Hellman," *J. Cryptol.*, vol. 17, no. 4, pp. 263–276, 2004.

[22] C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay, "Practical functional encryption for quadratic functions with applications to predicate encryption," in *Proc. Annu. Int. Cryptol. Conf.*, 2017, pp. 67–98.

[23] A. A. Badawi et al., "The AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs," 2018, *arXiv:1811.00778*.

[24] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Proc. Annu. Int. Cryptol. Conf.*, 2018, pp. 483–512.

[25] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2016, pp. 3–33.

[26] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.

[27] W. Hart, F. Johansson, and S. Pancratz. "FLINT: Fast library for number theory, V. 2.4.3." 2013. [Online]. Available: http://flintlib.org

[28] J. A. Akinyele, M. D. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, 2013.

[29] B. Lynn, *PBC Library Manual 0.5.11*, Stanford Univ., Stanford, CA, USA, 2006.

[30] E. D. Sans, R. Gay, and D. Pointcheval, "Reading in the dark: Classifying encrypted digits with functional encryption," IACR, Bellevue, WA, USA, Rep. 206/2018, 2018.

[31] S. Carpov, C. Fontaine, D. Ligier, and R. Sirdey, "Illuminating the dark or how to recover what should not be seen in FE-based classifiers," in *Proc. Privacy Enhanc. Technol.*, 2020, pp. 5–23.

[32] S. Agrawal, B. Libert, and D. Stehlé, "Fully secure functional encryption for inner products, from standard assumptions," in *Proc. Annu. Int. Cryptol. Conf.*, 2016, pp. 333–362.

[33] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1322–1333.

[34] Y. LeCun, C. Cortes, and C. J. Burges, (ATT Labs, Atlanta, GA, USA). *MNIST Handwritten Digit Database: Volume 2*. (2010). [Online]. Available: http://yann.lecun.com/exdb/mnist

[35] R. Kohavi, "Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid," in *Proc. KDD*, vol. 96, 1996, pp. 202–207.

[36] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[37] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2015, *arXiv:1603.04467*.

[38] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011.

[39] F. Chollet et al., "Keras: The Python deep learning library," *Astrophys. Source Code Library*, 2018. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2018ascl.soft06022C/abstract

[40] S. Agrawal, B. Libert, M. Maitra, and R. Titiu, "Adaptive simulation security for inner product functional encryption," IACR, Bellevue, WA, USA, Rep. 2020/209, 2020. [Online]. Available: https://eprint.iacr.org/2020/209

[41] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library—SEAL V2.1," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2017, pp. 3–18.

[42] S. Halevi and V. Shoup, "Algorithms in HElib," in *Proc. Annu. Cryptol. Conf.*, 2014, pp. 554–571.

[43] M. Abdalla, R. Gay, M. Raykova, and H. Wee, "Multi-input inner-product functional encryption from pairings," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Technol.*, 2017, pp. 601–626.

[44] Z. Brakerski, I. Komargodski, and G. Segev, "Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions," *J. Cryptol.*, vol. 31, no. 2, pp. 434–520, 2018.

[45] M. Ambrona, D. Fiore, and C. Soriente, "Controlled functional encryption revisited: Multi-authority extensions and efficient schemes for quadratic functions," in *Proc. Privacy Enhanc. Technol.*, 2021, pp. 21–42.

[46] J. Chotard, E. D. Sans, R. Gay, D. H. Phan, and D. Pointcheval, "Decentralized multi-client functional encryption for inner product," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2018, pp. 703–732.

[47] J. Chotard, E. Dufour-Sans, R. Gay, D. H. Phan, and D. Pointcheval, "Dynamic decentralized functional encryption," in *Proc. Annu. Int. Cryptol. Conf.*, 2020, pp. 747–775.

[48] M. Bahadori and K. Järvinen, "A programmable SoC-based accelerator for privacy-enhancing technologies and functional encryption," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 10, pp. 2182–2195, Oct. 2020.

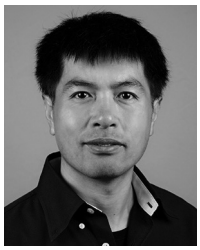
Prajwal Panzade (Member, IEEE) received the master's degree in computer science and engineering (focused on information security) from the Indian Institute of Technology Dhanbad, Dhanbad, India, in 2017. He is currently pursuing the Ph.D. degree with the Department of Computer Science, INSPIRE Center, Georgia State University, Atlanta, GA, USA.

He was a Lecturer with the National Institute of Technology Andhra Pradesh, Tadepalligudem, India, from 2017 to 2019. His research interests include privacy-preserving machine learning, applied cryptography, digital image forensics, machine learning, and federated learning.

Daniel Takabi (Member, IEEE) received the Ph.D. degree in information science and technology from the University of Pittsburgh, Pittsburgh, PA, USA, in 2013.

He is currently a Professor and the Director of the School of Cybersecurity and a Batten Endowed Chair of Cybersecurity with Old Dominion University, Norfolk, VA, USA. Prior to this, he was the Founding Director of the Information Security and Privacy: Interdisciplinary Research and Education (INSPIRE) Center [designated as the National Center of Academic Excellence in Cyber Defense Research (CAE-R)], Georgia State University, Atlanta, GA, USA. His research interests include various aspects of cybersecurity and privacy, including trustworthy AI, privacy-preserving machine learning, adversarial learning, advanced access control models, insider threats, and usable security and privacy.

Dr. Takabi has served as a Technical Program/Organizing Committee Member for a number of conferences and workshops, including IEEE S&P, ACM CCS, ACSAC, ACM CODASPY, ACM SACMAT, and PETS.



Zhipeng Cai (Fellow, IEEE) received the B.S. degree from Beijing Institute of Technology, Beijing, China, in 2001, the master's degree in 2004 and the Ph.D. degree from the Department of Computing Science, University of Alberta, Edmonton, AB, Canada, in 2008.

He is currently a Professor with the Department of Computer Science, Georgia State University, Atlanta, GA, USA. His research has received funding from multiple academic and industrial sponsors, including the National Science Foundation and the U.S. Department of State, and has resulted in over 100 publications in top journals and conferences, with more than 14 500 citations, including over 80 IEEE/ACM transactions papers. His research expertise lies in the areas of resource management and scheduling, privacy, networking, and big data.

Dr. Cai is the recipient of an NSF CAREER Award. He is the Editor-in-Chief of *Wireless Communications and Mobile Computing*, an Associate Editor-in-Chief of *High-Confidence Computing* (Elsevier), as well as an Editor of various prestigious journals, such as IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS.