Privacy-Preserving Data Evaluation via Functional Encryption, Revisited

Xinyuan Qian[1], Hongwei Li[1], Guowen Xu[1,5], Haoyong Wang[2], Tianwei Zhang[3], Xianhao Chen[4], Yuguang Fang[5]
[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China
[2] Chongqing Key Laboratory of Automated Reasoning and Cognition
[3] School of Computer Science and Engineering, Nanyang Technological University
[4] Department of Electrical and Electronic Engineering, University of Hong Kong
[5] Department of Computer Science, City University of Hong Kong

***Abstract*—In cloud-based data marketplaces, the cardinal objective lies in facilitating interactions between data shoppers and sellers. This engagement allows shoppers to augment their internal datasets with external data, consequently leading to significant enhancements in their machine learning models. Nonetheless, given the potential diversity of data values, it becomes critical for consumers to assess the value of data before cementing any transactions. Recently, Song et al. introduced Primal (publish in ACSAC), the pioneering cloud-assisted privacy-preserving data evaluation (PPDE) strategy. This strategy relies on variants of functional encryption (FE) as the underlying framework, conferring notable performance advantages over alternative cryptographic primitives such as secure multi-party computation and homomorphic encryption. However, in this paper, we regretfully highlight that Primal is susceptible to inadvertent misuse of FE, and leaves much-desired room for performance amelioration. To combat this, we introduce a novel cryptographic primitive known as labeled function-hiding inner-product encrypted. This new primitive serves as a remedy and forms the foundation for designing the concrete framework for PPDE. Furthermore, experiments conducted on real datasets demonstrate that our framework significantly reduces the overall computation cost of the current state-of-the-art secure PPDE scheme by roughly 10× and the communication cost for the data seller by about 2×.**

***Index Terms*—Privacy-Preserving, Data Evaluation, Functional Encryption**

I. Introduction

A. Data Evaluation (DE)

Deep learning techniques have greatly improved traditional machine learning (ML) in applications like image recognition [1], [2], medical prediction [3], [4], and autopilot [5]. These algorithms require ample high-quality data [6] for training effective models. To meet the demand for additional data, data marketplaces have emerged [7], [8]. These platforms operate on a Data-as-a-Service model [9], facilitating the exchange and access of diverse datasets on a large scale for enterprises and individuals.

Corresponding Author: Hongwei Li, Email: *hongweili@uestc.edu.cn*.

This work is supported by the National Key R&D Program of China under Grant 2022YFB3103500, the National Natural Science Foundation of China under Grants 62020106013, the Sichuan Science and Technology Program under Grants 2023ZYD0142, the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027 and Y030232063003002.

A data marketplace not only provides a diverse range of data for training better ML models but also offers cloud-based data evaluation services to assist shoppers in assessing and acquiring high-quality data [10], resulting in significant enhancements in model performance.

Traditionally, the evaluation of ML data for shoppers necessitates cloud access to both the shopper's model and the sellers' data. However, sellers are hesitant to expose their data prior to receiving payment due to concerns about potential data leaks and devaluation. Likewise, shoppers are unwilling to disclose their proprietary models as they are valuable digital assets. Consequently, the development of a privacy-preserving data evaluation (PPDE) framework is imperative in a cloud-based data marketplace.

Existing cloud-based data marketplaces [11], [12] often have a limited focus, either supporting data evaluation for machine learning models or providing privacy protection solely for models, data, and services in the cloud. Few platforms successfully integrate both aspects. While current machine learning privacy-preserving schemes [3], [13]–[17] can safeguard the privacy of models and data in the cloud, they often result in significant computational or communication overhead, particularly when executing model operations like predicting data under ciphertext. Moreover, these schemes, whether utilizing homomorphic encryption (HE) [13]–[15] or multiparty computation (MPC) [3], [16], [17], either only perform feed-forward computations to compute predicted values in a secure manner (HE-based schemes) or rely on a multi-server setup (MPC-based schemes), which suffers from server collusion and is difficult to realize in the real world, rendering them impractical for tasks that necessitate privacy-preserving data evaluation.

B. PPDE via FE: State of the Art

To the best of our knowledge, Song et al. present the first end-to-end method [18] (call Primal published in ACSAC) specifically tailored for privacy-preserving data evaluation (PPDE) thus far. This method utilizes lightweight functional encryption (FE) as its foundation and introduces a customized inner product FE strategy to enhance performance. Experimental results vividly illustrate the superior computational perfor-

mance of Primal compared to other cryptographic primitives such as MPC and HE.

However, we regret to highlight that Primal is not as secure as claimed by the authors, primarily due to the improper utilization of FE (see section IV for more information). Furthermore, the communication and computing overhead of Primal present a significant gap in practicality, leaving much-desired room for performance improvement. As a result, it is essential to thoroughly address and rectify these vulnerabilities in order to strengthen the scheme's resilience and safeguard the privacy of the evaluated data.

C. Our Contributions

In this study, our objective is to highlight the significant disparity in the practicality of current research, particularly in the case of Primal-based PPDEs, concerning security and efficiency. Furthermore, we present a privacy-enhancing alternative that offer satisfactory performance. To the best of our knowledge, our alternative, an extension of Primal, is essentially the first implementation of PPDE that excels in being lightweight while ensuring semantic security. In summary, our contributions can be outlined as follows.

(1) Vulnerability of Primal. We discover critical flaws in Primal, specifically in its misuse of inner-product FE (IPFE), which inadvertently enables the cloud to infer the seller's raw data. Put simply, a passive adversary can execute the following three types of attacks through meticulous operation.

A1. Exploiting simultaneous equations via Gaussian elimination to ascertain the seller's input data. **A2.** Colluding with the shopper and launch a "Mix-and-Match" attack to train the encrypted ML model without rendering the necessary payment. **A3.** Inferring the original data by utilizing the master secret key. It is important to highlight that such information leakage occurs even when employing a provably secure IPFE scheme, such as the one used in [18] based on [19].

(2) New Variant of Function Encryption. We introduce an efficient and practical cryptographic tool, labeled function-hiding inner-product encryption (labeled-FHIPE) to remedy the weakness in Primal.

Our first innovation involves a label-matching mechanism in the novel labeled-FHIPE scheme. It guarantees decryption and private function computation only when decryption key and ciphertext labels match, effectively addressing "Mix-and-Match" (**A2**) attacks. This control over ciphertext access ensures precise use of encrypted data for our computational tasks. The second innovation harnesses function-hiding capability of labeled-FHIPE to re-encrypt and hide the shopper's model, addressing data leakage inherent in the Primal approach, effectively countering **A1** attacks caused by intermediate data exposure. Finally, our privacy-preserving framework, built on labeled-FHIPE, includes a Trusted Third Party (TTP) for secure master secret key management. This essential incorporation, a standard requirement for Functional Encryption (FE) schemes [20], [21] except some decentralized designs, effectively addresses vulnerabilities in **A3**. By protecting the master secret key from both sellers and shoppers, our framework averts **A3**-type attacks, preventing direct inference of sensitive information.

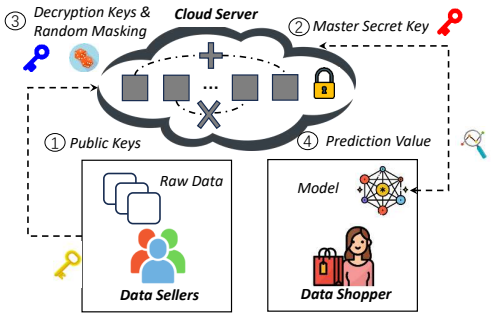


Fig. 1. PPDE Workflow

(3) Experimental Analysis. We perform an evaluation using three datasets and two unique neural network architectures to measure computational and communication overhead, and to assess the effectiveness of data selection and validation. The results exhibit that our framework significantly reduces the overall computation cost of the current best secure PPDE scheme by roughly $10\times$ and the communication cost for the data seller by about $2\times$. Moreover, our approach particularly excels in scalability, as demonstrated by the linear growth of computation time in the first-layer private inner-product calculation with increasing vector dimensions or model precision.

II. Privacy-Preserving Data Evaluation

A. Neural Networks and Pre-Process for Convolutional Layer

Traditional neural networks process data through interconnected layers, including input, hidden, and output layers. The hidden layers, often fully-connected or convolutional, allow complex pattern learning.

In a convolutional layer, necessary operations are carried out between inputs from the preceding layer and the current layer's convolutional kernels. Effectively, these can be construed as inner product operations. For the sake of simplicity in the subsequent descriptions, we introduce a data preprocessing approach that recasts convolutional operations into inner product operations. Initially, the i-th layer's convolutional kernel $\mathbf{K}_i^{m\times n}$ is converted into a vector $\mathbf{k}_i$ of length $m \cdot n$. For input data, we restructure each layer's feature maps into a matrix $\mathbf{T}_i^{m'\times n'}$, where $n' = m \times n$, as per the actual stride. As a result, the convolutional layer's output feature map can be seen as a matrix multiplication $\mathbf{z}_i = \mathbf{T}_i \times \mathbf{k}_i$. This vector $\mathbf{z}_i$ is then reshaped into a matrix $\mathbf{Z}_i$ for output.

B. Privacy-Preserving Data Evaluation

In the data marketplace, sellers strive to provide high-quality data while shoppers, leveraging cloud services, evaluate this data to inform purchasing decisions. The evaluation process often necessitates data disclosure, raising privacy concerns. To mitigate these, Privacy-Preserving Data Evaluation (PPDE) techniques are proposed.

The PPDE workflow involves sellers encrypting their data for cloud upload, and shoppers doing likewise with their models. The cloud server processes the encrypted data, enabling data selection and validation. In the *data selection* phase, the shopper, upon receiving the computed encrypted predictions from the cloud server, decrypts it and applies active learning to select quality data. During the *data validation* phase, the encrypted data is utilized to train the fully encrypted network. After multiple training rounds, the data quality is evaluated by observing the predicted performance.

C. Active Learning in PPDE

Active learning, a key component of PPDE, enables the selection of informative training data from an unlabeled pool, enhancing data quality. The cloud server calculates on encrypted data and the model to output an encrypted prediction value. Once decrypted, this value aids the shopper in assessing the training data quality via active learning, minimizing the need for extensive labeled data.

Active learning smartly chooses beneficial data for labeling, often using uncertainty measures. For example, uncertainty measure selection in [22] uses the prediction values to pick informative data, bypassing the need for direct data access. Integrating active learning with privacy-preserving techniques ensures efficient, privacy-preserving data evaluation, which is vital for sensitive domain applications.

III. Cryptographic Background

A. Notations

Here, $\mathbb{Z}$ symbolizes the set of integers, while $\mathbb{Z}_p$ signifies the quotient ring of integers modulo a positive integer q. Bold lowercase letters like $\mathbf{x}$ represent vectors. The cardinality of a finite set $\mathcal{X}$ is represented as $|\mathcal{X}|$. $a \leftarrow \mathcal{A}$ implies a is the output of algorithm $\mathcal{A}$ if $\mathcal{A}$ is an algorithm; otherwise, a is a uniformly sampled element from set $\mathcal{A}$. The hadamard product between two matrices or vectors of identical dimensions is denoted by $\circ$. The dimension of a variable is signified by $|\cdot|$. $\mathbf{A}[i]$ refers to the i-th row in matrix $\mathbf{A}$, while $\mathbf{A}[i][j]$ represents the element at the intersection of the i-th row and j-th column.

B. Functional Encryption for Inner Product: A Sketch

We first briefly introduce functional encryption for inner product (IPFE). An IPFE scheme consists of four Probabilistic Polynomial Time (PPT) algorithms:

- $\mathsf{Setup}(1^\kappa, 1^l) \rightarrow (\mathsf{mpk}, \mathsf{msk})$: It takes the security parameter 1^κ and dimension 1^l as input, and outputs master public/secret key pair $(\mathsf{mpk}, \mathsf{msk})$;
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}) \rightarrow \mathsf{dk}_\mathbf{y}$: It takes a master secret key msk, a weight vector $\mathbf{y} \in \mathbb{Z}_p^l$ as input, and outputs decryption key $\mathsf{dk}_\mathbf{y}$;
- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}) \rightarrow \mathsf{ct}_\mathbf{x}$: It takes a master public key mpk, a message $\mathbf{x} \in \mathbb{Z}_p^l$ as input, and outputs a ciphertext $\mathsf{ct}_\mathbf{x}$;
- $\mathsf{Dec}(\mathsf{dk}_\mathbf{y}, \mathsf{ct}_\mathbf{x}, \mathbf{y}) \rightarrow z$: It takes a decryption key $\mathsf{dk}_\mathbf{y}$, a ciphertext $\mathsf{ct}_\mathbf{x}$, and a weight vector $\mathbf{y}$ as input, and outputs the original inner product result z.

Due to space limitations, please refer to [19] for the definitions of correctness and security for the IPFE scheme.

$\mathsf{Real}_{\mathcal{A}}^{\mathsf{CPA\text{-}FE}}(1^\kappa)$	$\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{CPA\text{-}FE}}(1^\kappa)$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, 1^l)$	$(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{msk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\kappa, 1^l)$
$\mathbf{x}^* \leftarrow \mathcal{A}(1^\kappa, \mathsf{pk})$	$\mathbf{x}^* \leftarrow \mathcal{A}(1^\kappa, \widetilde{\mathsf{pk}})$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathbf{x}^*)$	$\mathsf{ct}^* \leftarrow \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{msk}})$
$\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot)}(\mathsf{ct}^*)$	$\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot)}(\mathsf{ct}^*)$

Fig. 2. Security Experment

C. Function-Hiding Inner Product Encryption

Function-Hiding Inner Product Encryption (FHIPE) enhances IPFE by concealing the weight information, thereby offering a more comprehensive privacy safeguard. FHIPE facilitates weight encryption, preventing unauthorized access to specific weight details. Consequently, FHIPE provides an advanced privacy-preserving measure, accommodating a wider application spectrum where privacy is paramount. We formally examine the four PPT FHIPE algorithms below [23]:

- $\mathsf{Setup}(1^\kappa, 1^l) \rightarrow (\mathsf{pk}, \mathsf{msk})$: It takes the security parameter 1^κ and dimension 1^l as input, and outputs the public key and the master secret key $(\mathsf{pk}, \mathsf{msk})$;
- $\mathsf{Enc}(\mathsf{pk}, \mathbf{x}) \rightarrow \mathsf{ct}_\mathbf{x}$: It takes a public key pk, a message $\mathbf{x} \in \mathbb{Z}_p^l$ as input, and outputs a ciphertext $\mathsf{ct}_\mathbf{x}$;
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}) \rightarrow \mathsf{dk}_\mathbf{y}$: It takes a master secret key msk and a weight vector $\mathbf{y} \in \mathbb{Z}_p^l$ as input, and outputs a decryption key $\mathsf{dk}_\mathbf{y}$.
- $\mathsf{Dec}(\mathsf{dk}_\mathbf{y}, \mathsf{ct}_\mathbf{x}) \rightarrow z$: It takes a master public key mpk, a decryption key $\mathsf{dk}_\mathbf{y}$ and a ciphertext $\mathsf{ct}_\mathbf{x}$ as input, and outputs the original inner product result z.

1) *Correctness:* For all $\kappa, l \in \mathbb{N}$, $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^l$, we require

$$\Pr[\mathsf{Dec}(\mathsf{dk}_\mathbf{y}, \mathsf{ct}_\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle] = 1 - \mathsf{negl}(\kappa),$$

where $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, 1^l)$, $\mathsf{dk}_\mathbf{y} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$, and $\mathsf{ct}_\mathbf{x} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{x})$. As state in [23], it requires the above holds when $\langle \mathbf{x}, \mathbf{y} \rangle \in B$ where $B \subseteq \mathbb{Z}_p$.

2) *Security:* As shown in Fig. 2, we define the simulation-based security via the following two experiments for any security parameter κ, any *PPT* simulator $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$, any *PPT* stateful adversary $\mathcal{A}$.

An FHIPE is *CPA-SIM* secure if there exists a *PPT* simulator $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$ such that for all *PPT* adversaries $\mathcal{A}$, we have

$$\begin{aligned}\mathsf{Adv}_{\mathsf{FHIPE},\mathcal{A}}^{\mathsf{CPA\text{-}SIM}}(\kappa) := |\Pr[1 \leftarrow \mathsf{Real}_{\mathcal{A}}^{\mathsf{CPA\text{-}FE}}(1^\kappa)] - \\ \Pr[1 \leftarrow \mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathsf{CPA\text{-}FE}}(1^\kappa)]| = \mathsf{negl}(\kappa)\end{aligned}$$

IV. Improper Utilization of IPFE in PPDE

This section provides an overview of a foundational work called Primal [18], which is based on PPDE via IPFE, and discusses its structural details. We identify security vulnerabilities due to IPFE's improper application within the PPDE context, leading to potential exploits by an honest-but-curious cloud or shopper, enabling the launch of three types of attacks:

- **A1**. exploitation of simultaneous equations through Gaussian elimination to deduce the seller's input data.

- **A2**. collusion between the cloud and the shopper to execute a "Mix-and-Match" attack, enabling the training of an encrypted Machine Learning (ML) model without the required payment.
- **A3**. inference of the original data by leveraging the master secret key.

Such attacks could expose more information about the seller's data and the shopper's model than a semi-honest cloud should access. We detail these attacks in Section IV-B to IV-D.

A. Review of PPDE via IPFE

The initial PPDE scheme primarily involves the following procedures (see Figure 1): The data seller encrypts the sale data using a public key and sends it to the cloud. The data shopper encrypts the model using random masking and generates decryption keys with the IPFE.KeyGen algorithm. The cloud server, using decryption keys and random masks, separately computes outputs of the first layer and subsequent layers of the model, eventually obtaining masked prediction results which are then sent to the shopper for unmasking.

B. Attack 1: Data Inference

In the PPDE via IPFE, the cloud could potentially infer the seller's private data through the first hidden layer computation, failing to uphold the security guarantees in a PPDE scenario. For example, when the first hidden layer is a fully-connected layer, the seller's data (x_1, x_2, x_3) encrypted by IPFE.Enc, and the layer $\mathbf{RW}_1$ disguised with random numbers is encrypted by IPFE.KeyGen. Although at first, it may seem the masked model is hidden in sk_i, the reality is different. After the IPFE decryption, the inner-product results are exposed, enabling the derivation of the seller's input data, as is shown in Fig. 3.

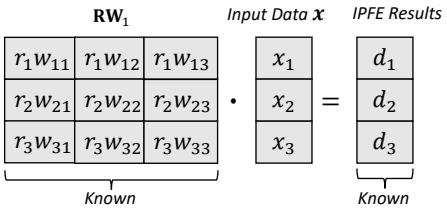


Fig. 3. A1 Attack

C. Attack 2: Collusive Training

The shopper and the cloud may make huge benefit by complicitly training the model without paying for the seller's data. Let us start from an example. The cloud has the seller's encrypted data and the shopper has the master secret key, which can be used to train any models with any encrypted data. The cloud can prepare two original models, one for privacy-preserving data evaluation and another for training the encrypted model. After data evaluation, the cloud can use the desired encrypted data for the shopper to training the other clean model under ciphertext. The model can be decrypted by the shopper once the training process is finished. This scenario allows the cloud and the shopper to bypass payment for the sellers' data.

D. Attack 3: Master Secret Key Misuse

Holding the master secret key msk, the shopper can infer or manipulate more of the sellers' data. The msk allows the generation of the decryption key $\mathsf{dk}_{\mathbf{y}}$ at will. By specifying the corresponding $\mathbf{y}$, the privacy of the data seller can be compromised through the computed inner product result. For example, $\mathbf{y} = (0, 0, ..., y_t = 1, ..., 0, 0)$, then the inner product result is $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i = x_t$. Such actions can lead to inference of the original user data.

In summary, the use of IPFE in PPDE cannot ensure the desired privacy due to the aforementioned potential attacks. Such information leakage could occur even with a provably secure IPFE scheme as all the decryption key and ciphertext manipulations are permitted by the functionality of IPFE.

V. Our Framework

In this section, we highlight the design challenges inherent to PPDE via IPFE, describe our threat model, and propose our solution to these issues. This includes a new FE variant termed labeled-FHIPE, which we apply to PPDE, and provides a detailed outline of this privacy-preserving structure.

A. Challenges

The innate properties of IPFE render it unsuitable for directly addressing the issues associated with PPDE. The specific challenges stem from the following factors:

- **Excessive Information Exposure in IPFE:** Due to the involvement of weight $\mathbf{y}$ in inner-product computations in the decryption algorithm, IPFE cannot achieve function-hiding or avoid weight leakage.
- **Permissiveness of "Mix-and-Match" in IPFE:** IPFE decryption can utilize mismatched decryption keys as long as the keys correspond to the same data length [19], a feature which can unintentionally expose data control in PPDE.
- **Centralized Control in IPFE:** Generally, IPFE necessitates a central TTP for key distribution and functional decryption key generation. Yet, current PPDE schemes [18] delegate the responsibility of master secret key generation and management to data shoppers, which, despite decentralizing control, invariably increases the shopper's attack potential.

B. Threat Model and Design Goals

In our study, we consider a typical cloud-based data marketplace model, comprising the following key entities: a Trusted Third Party (TTP), semi-honest data sellers, potentially colluding data shoppers, and an untrusted, though honest-but-curious, cloud.

TTP, assumed to be reliable and non-colluding, holds the master key pair and oversees key distribution and function-derived secret key generation. The TTP model is widely recognized as the underlying infrastructure in numerous existing cryptosystems [21], [24], especially for FE-based systems [20], [25]. Data sellers upload their data to the cloud for sale, seeking maximum profit without launching data poisoning attacks, though they may unintentionally possess some irrelevant or mislabeled data [18]. The data shopper aims

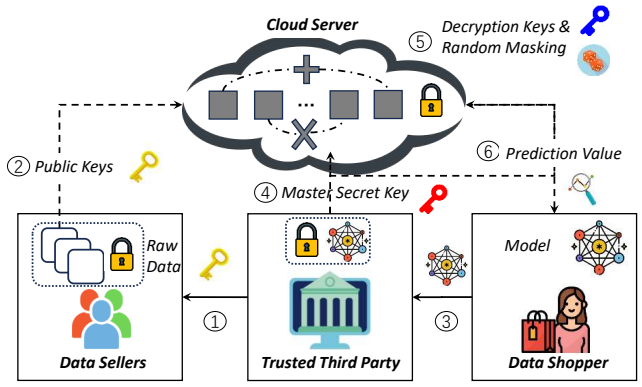


Fig. 4. Privacy-Preserving Data Evaluation Architecture

to acquire valuable sellers' data from the cloud to enhance their machine learning model, retrieving auxiliary information, like prediction values, for data evaluation, and may collude with the cloud to train the model without purchasing the seller's private data. The cloud, assumed to be honest-but-curious, adheres to the predefined protocol but may exploit the shopper's model and sellers' data for profit, and could also collude with the data shopper to train a model without compensating sellers.

Our central goal is to address the security vulnerabilities in the extant leading PPDE scheme and develop a novel PPDE framework capable of ensuring robust security even under potential collusions between the cloud and shoppers.

C. Method Overview

In order to address the aforementioned objectives, we propose the following innovative strategies as follows:

New Variant of FE. We build upon FHIPE, which aptly addresses **A1** by allowing the cloud to calculate the first layer output without needing knowledge of the encrypted model parameters $\mathbf{y}$. To tackle **A2**, we introduce a novel FE variant known as labeled-FHIPE. In this scheme, the seller encrypts their valuable data into a ciphertext $\mathsf{ct}_{\mathbf{x},\ell}$ using a label ℓ. Concurrently, TTP generates a label-bound decryption key $\mathsf{dk}_{\mathbf{y},\ell}$. Only one with a ciphertext and decryption key with matching labels can compute the inner product. Section V-D offers a specific construction for the labeled-FHIPE.

Enhanced PPDE. Utilizing labeled-FHIPE, we put forth a new PPDE framework that engages TTP for managing the master secret key. This addresses **A3** by restricting the data shopper's information access and reducing their potential attack capability. It is important to note that our framework retains all advantages offered by the original PPDE. The specifics of our scheme will be elaborated on in Section V-E.

D. Labeled-FHIPE

Here we presents a labeled-FHIPE scheme from SXDH assumption under random oracle model. We start from FHIPE where the seller encrypts data $\mathbf{x}$; TTP issues a key for $\mathbf{y}$; decryption recovers $z = \langle \mathbf{x}, \mathbf{y} \rangle$. To further prevent the decryptor (i.e., the cloud) colluding with shoppers and launch a mix-and-match attack, we let the seller embed a fresh label $h := \mathcal{H}(l) \cdot \mathsf{sk} \in \mathbb{G}$ into its ciphertext so that only decryption key with the same label can decrypt the ciphertext correctly, where $\mathsf{sk} \in \mathbb{Z}_p$ is a random secret key and $\mathcal{H}(\cdot) := \mathsf{Label} \to \mathbb{G}$. In particular, the data seller encrypts $\mathbf{x} \| h$ instead of $\mathbf{x}$; TTP issues a key for $\mathbf{y} \| h^{-1}$ instead of $\mathbf{y}$; decryption obtains $z' = \langle \mathbf{x} \| h, \mathbf{y} \| h^{-1} \rangle - 1 = \langle \mathbf{x}, \mathbf{y} \rangle$. In the proof, we will use h as a label vector that binds $\mathbf{x}$ and $\mathbf{y}$; for this, we also introduce one more dimension for the security proof.

1) Algorithms: Let $(\mathsf{Setup}_0, \mathsf{KeyGen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$ be the FHIPE scheme in [23], our labeled-FHIPE is as follows:

- $\mathsf{Setup}(1^\kappa, \mathcal{PG}, [\mathbf{M}]_1)$: Output $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_p$ and $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}_0(1^\kappa, \mathcal{PG}, [\mathbf{M}]_1)$.
- $\mathsf{Enc}(\mathbf{t} \in \mathbb{Z}_p^m, l \in \ell)$: Output $\mathsf{ct}_{\mathbf{Mt},l} \leftarrow \mathsf{Enc}_0(\mathbf{t}_l \in \mathbb{Z}_p^m)$. All intermediate results $\mathbf{Mt} = \mathbf{x} \in \mathbb{Z}_p^n$ are replaced by $\mathbf{x} \| h_l \in \mathbb{Z}_p^{n+1}$, where $h_l := \mathcal{H}(l) \cdot \mathsf{sk} \in \mathbb{G}$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y} \in \mathbb{Z}_p^n, l \in \ell) \to \mathsf{dk}_{\mathbf{y},l}$: Output $\mathsf{dk}_{\mathbf{y},l} \leftarrow \mathsf{KeyGen}_0(\mathsf{msk}, \mathbf{y} \| h_l^{-1} \in \mathbb{Z}_p^{n+1})$, $h_l := \mathcal{H}(l) \cdot \mathsf{sk} \in \mathbb{G}$.
- $\mathsf{Dec}(\mathsf{ct}_{\mathbf{x},l}, \mathsf{dk}_{\mathbf{y},l})$: Compute $z_0 \leftarrow \mathsf{Dec}_0(\mathsf{ct}_{\mathbf{x},l}, \mathsf{dk}_{\mathbf{y},l})$, and output $z = z_0 - 1$.

2) Label Correctness: Because $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^n$ are bound to h and h^{-1}, correct decryption is only possible when $\mathbf{x}$ is labeled with the inverse of the label of $\mathbf{y}$: $z' = \langle \mathbf{x} \| h, \mathbf{y} \| h^{-1} \rangle - 1 = \langle \mathbf{x}, \mathbf{y} \rangle$. This immediately verifies the label's correctness.

3) Label Security: We prove the following theorem by showing that our proposed labeled-FHIPE scheme achieves label security (LS). Technically, the theorem shows that LS relies on the security of the underlying FHIPE scheme. By using the scheme from [23], we obtain a scheme with LS based on the SXDH assumption under random oracle model.

Theorem 1: Assume $(\mathsf{Setup}_0, \mathsf{KeyGen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$ is a secure FHIPE. Our scheme achieves semi-adaptive label security, where the adversary sends its challenge $\mathbf{x}$ before querying any secret keys, but after receiving the public key. Formally, for all p.p.t. adversary $\mathcal{A}$, there exists a p.p.t. adversary $\mathcal{B}$ with

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Labeled-FHIPE}}(\kappa) \leq \mathsf{Adv}_{\mathcal{B}}^{\text{FHIPE}}(\kappa).$$

Proof 1: We prove the theorem via a game sequence $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$ with $\mathbf{G}_0 \approx_c \mathbf{G}_1 \approx_c \mathbf{G}_2$. Each step is standard, and we only present games and sketch the proof for indistinguishability between two adjacent games. In the following, we use $\mathsf{Adv}_{\mathcal{A}}^i$ to denote the advantage of $\mathcal{A}$ in $\mathbf{G}_i$ for all $i \in \{0, 1, 2\}$.

$\underline{\mathbf{G}_0}$: This is the real game with $b \leftarrow \{0, 1\}$ being the secret bit. Apart from pk, adversary $\mathcal{A}$ gets challenge ciphertext:

$$\mathsf{ct}_{\mathbf{Mt},l} \leftarrow \mathsf{Enc}_0(\mathsf{pk}, \mathbf{t}_l), \quad \forall l,$$

, where $h_l := \mathcal{H}(l) \in \mathbb{G}$ and $\mathbf{Mt}_l = \mathbf{x} \| h_l$, and a decryption key for single function $\mathbf{y}$:

$$\mathsf{dk}_{\mathbf{y},l} \leftarrow \mathsf{KeyGen}_0(\mathsf{msk}, \mathbf{y} \| h_l^{-1}), \quad \forall l.$$

We can get $\mathsf{Adv}_{\mathcal{A}}^{\text{Labeled-FHIPE}}(\kappa) = \mathsf{Adv}_{\mathcal{A}}^0$ for all $\mathcal{A}$.

$\underline{\mathbf{G}_1}$: This game is identical to $\mathbf{G}_0$ except that we replace the hash function $\mathcal{H}$ that is evaluated in every random-oracle query l, with a random function $\mathsf{RF} := z \leftarrow_\$ \mathbb{Z}_p$. We have $\mathbf{G}_1 \approx_c \mathbf{G}_0$ and $\mathsf{Adv}_{\mathcal{A}}^1(\kappa) = \mathsf{Adv}_{\mathcal{A}}^0(\kappa)$.

Initialization phase: TTP runs Setup algorithm of labeled-FHIPE to generate a master secret key msk for the system and a pair of public and secret key $(\mathsf{pk}, \mathsf{sk})$ for a data seller. Then the shopper randomly chooses numbers $\mathbf{R}_i$ from corresponding linear space associated with i-th layer, where each coefficient of $\mathbf{R}_i$ are from $\mathbb{Z}_p$, $i \in \{1, 2, ..., n\}$ and n is the total layer number. To be specific, we use $\mathbf{R}_i$ to denote vector or matrix of random numbers used to mask the i-th fully-connected hidden layer and the i-th convolutional hidden layer, respectively.

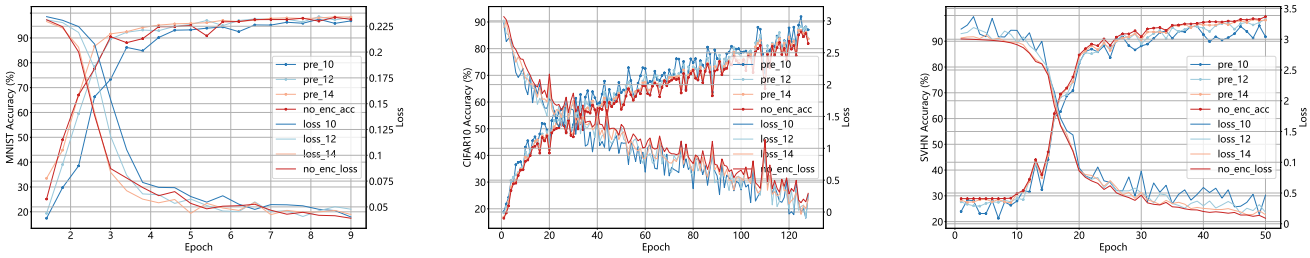
Feed Forward phase:

- **Data Encryption:**
  - Process a seller's raw data into $\mathbf{X}$ according to the type of first hiding layer. If it is the fully-connected hidden layer, $\mathbf{X}$ denotes a $1 \times n_1$-dimension matrix . If it is the convolutional hidden layer, it represents $m_1 \times n_1$ matrix after convolutional layer data processing (see Section II-A).
  - For each row $\mathbf{X}[i]$ in $\mathbf{X}$, the data seller encrypts it as $\mathsf{ct}_{\mathbf{X}[i],l} \leftarrow$ labeled-FHIPE.$\mathsf{Enc}(\mathbf{X}[i], l)$, where $l = \mathsf{id}\|\mathsf{task}\|\mathsf{stp}$. Then the data seller sends every $\mathsf{ct}_{\mathbf{X}[i],l}$ to the cloud.
- **First Layer Encryption:**
  - The data shopper masks the first hidden layer. For fully-connected hidden layer, masks parameter matrix $\mathbf{W}_1$ row by row $\mathbf{Y}_1[i] = \mathbf{R}_1[i] \circ \mathbf{W}_1[i]$, where $\mathbf{R}_1[i] = (r_1[i], r_1[i], ...)$ and $r_1[i] \leftarrow_{\$} \mathbb{Z}_p$. For convolutional hidden layer, mask the kernel vector $\mathbf{k}_1$ by $\mathbf{Y}_1[i * t + j] = \mathbf{R}_1[i][j] * \mathbf{k}_1$, where $t = (I - |\mathbf{K}'_1| + 1)/s$, $\mathbf{R}_1 \in \mathbb{Z}_p^{t \times t}$, I is the width of input, $\mathbf{K}'_1$ is the kernel matrix before data pre-processing, and s is the stride. The data shopper sends $\mathbf{Y}_1$ to TTP.
  - The TTP hides the masked model and generates decryption keys. For each row $\mathbf{Y}_1[i]$ in $\mathbf{Y}_1$, TTP computes $\mathsf{dk}_{\mathbf{Y}_1[i],l} \leftarrow$ labeled-FHIPE.$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{Y}_1[i], l)$ and sends them to the cloud.
- **First Layer Computation:**
  - The cloud parse the ciphertexts and decryption keys as $(\mathsf{ct}_{\mathbf{X}_i,l}, \mathsf{dk}_{\mathbf{Y}_1[i],l})_{i \in [|\mathbf{Y}_1|]}$. It computes the inner-product results $\mathbf{z}_1[i] \leftarrow$ labeled-FHIPE.$\mathsf{Dec}(\mathsf{ct}_{\mathbf{X}_i,l}, \mathsf{dk}_{\mathbf{Y}_1[i],l})$ for $i \in |\mathbf{Y}_1|$.
  - If it is the fully-connected hidden layer, we use $\mathbf{Z}_1 = \mathbf{z}_1^\top$ to denote a matrix with one row. If it is the convolutional hidden layer, rearrange $\mathbf{z}_1$ into matrix $\mathbf{Z}_1$ according to Section II-A. The cloud outputs square activation $\mathbf{Z}_1^2 = \mathbf{Z}_1 \circ \mathbf{Z}_1$ to the next layer.
- **Matrix Computation for the Subsequent i-th Layer:**
  - The cloud takes previous (i-1)-th layer output $\mathbf{Z}_{i-1}^2$ as input and do matrix computation according to layer type:
    - *Fully-connected hidden layer*. The cloud flattens $\mathbf{Z}_{i-1}$ into a vector if it is a matrix. Then, it receives the masked parameter matrix $\mathbf{RW}_i = \mathbf{W}_i \circ \mathbf{R}'_i$ from the shopper, where $\mathbf{R}'_i[u][v] = \mathbf{r}_i[u]/\mathbf{r}_{i-1}^2[v]$. Finally, it does matrix computation $\mathbf{Z}_i = \mathbf{RW}_i \times \mathbf{Z}_{i-1}$.
    - *Convolutional hidden layer*. For each row vector $\mathbf{Z}_{i-1}[j]$ in $\mathbf{Z}_{i-1}$, masks the kernel by $\mathbf{RK}_i[j] = \mathbf{R}'_i[j] \circ \mathbf{k}_i$, where $\mathbf{k}_i$ is kernel vector of the i-th layer and $\mathbf{R}'_i[u * |\mathbf{R}_i| + v] = \mathbf{R}_i[u][v]/\mathbf{R}_{i-1}[u * s : u * s + |\mathbf{K}_i|][v * s : v * s + |\mathbf{K}_i|]$, and computes $\mathbf{Z}_i[j] = \langle \mathbf{RK}_i[j], \mathbf{Z}_{i-1}[j] \rangle$.
  - The cloud outputs the square activation $\mathbf{Z}_i^2 = \mathbf{Z}_i \circ \mathbf{Z}_i$ to the next layer.
- **Prediction value output:**
  - If it is the last layer (the n-th layer), the cloud sends $\mathbf{Z}_n^2$ to the data shopper.
  - The data shopper receives $\mathbf{Z}_n^2$ and removes the mask by $\mathbf{Z}_n^2 \circ \mathbf{R}_n^{-2}$ to get the prediction value.

Back Propagation phase:

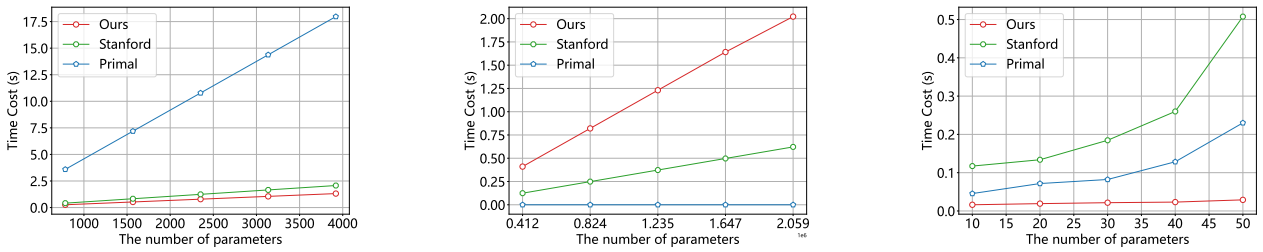
- **Back hidden layers computation:**
  - The shopper generates $\mathbf{M}_i = \mathbf{R}_i/\mathbf{T}_i$, where $\mathbf{T}_i$ is a random matrix, and sends it to the cloud.
  - The cloud updates parameters by $\mathbf{RW}'_i = \mathbf{RW}_i - \alpha * \mathbf{M}_i \times \frac{\partial \delta'}{\partial \mathbf{RW}_i} = \mathbf{R}'_i \times (\mathbf{W}_i - \alpha * \frac{\partial \delta}{\partial \mathbf{W}_i})$, where $\delta = \mathbf{Z}_n^2 - \mathbf{L}$, $\mathbf{L}$ denotes the labels, α is the learning rate, $\frac{\partial \delta'}{\partial \mathbf{RW}_i} = \mathbf{T}_i \times \frac{\partial \delta}{\partial \mathbf{W}_i}$ according to the chain rule. Here we randomize the gradients by computing $\delta' = \mathbf{R}_r \circ \delta$, where $\mathbf{R}_r$ represents a random vector $(r_1, r_2, ...)$.
- **First hidden layer computation:**
  - The cloud sends the gradients $\frac{\partial \delta'}{\partial \mathbf{Z}_1}$ to the seller.
  - The shopper sends the update parameter matrix $\mathbf{M}_1 = \mathbf{R}_1/\mathbf{T}_1$ to the seller.
  - The seller computes $\nabla \mathbf{Y}_1 = \mathbf{M}_1 \times \frac{\partial \delta'}{\partial \mathbf{Y}_1}$ and $\nabla \mathbf{Z}_1^2 = (\alpha * \nabla \mathbf{Y}_1 \mathbf{X}')^2$, where $\frac{\partial \delta'}{\partial \mathbf{Y}_1} = (\mathbf{X} \frac{\delta'}{\partial \mathbf{Z}_1})^\top = \mathbf{T}_1 \times \frac{\partial \delta}{\partial \mathbf{W}_1}$.
  - The seller updates the output $\mathbf{Z}_1^2$ using $\nabla \mathbf{Z}_1^2$ and starts a new training round.

Fig. 5. Our Data Evaluation Framework



(a) Mnist (b) Cifar10 (c) SVHN

Fig. 6. Influence of Precision on Accuracy



(a) Data Enc. (b) First Layer Enc. (c) First Layer Com.

Fig. 7. Comparison of Computational Overhead for Data Encryption, Model Encryption, First Layer Computation

$\underline{\mathbf{G}_2}$: This game is identical to $\mathbf{G}_1$ except that we answer the random-oracle queries for the label $l \neq l^*$ with an element indistinguishable from a random element. In particular, the random-oracle outputs a random number $z = a \cdot r$ with a random number $r \leftarrow_{\$} \mathbb{Z}_p$. This covers the results $\langle \mathbf{x}^b, \mathbf{y} \rangle + h_l \cdot z^{-1}$, which results in: $\mathsf{Adv}^2_{\mathcal{A}}(\kappa) - \mathsf{Adv}^1_{\mathcal{A}}(\kappa) \leq \mathsf{Adv}^{\mathrm{FHIPE}}_{\mathcal{B}}(\kappa)$, and we have $\mathbf{G}_2 \approx_c \mathbf{G}_1$.

Observe that in the final game $\mathbf{G}_2$, all challenge ciphertexts $\{\langle \mathbf{x}^b, \mathbf{y} \rangle + h_l \cdot z^{-1}\}_l \approx_c \mathcal{R}_p$ do not use the secret random bit b, where $\mathcal{R}_p$ is a random distribution over p; namely, the distribution of $\mathbf{G}_2$ is independent of b. Formally, for all $\mathcal{A}$, we have $Adv^2_{\mathcal{A}}(\kappa) = 0$. This readily proves the theorem. □

E. PPDE from FHIPE

In this part, we apply our labeled-FHIPE to PPDE and introduce our specific method. To address **A1** and **A2**, we use the labeled version of FHIPE to encrypt the data for sellers, so that every seller's ciphertext is labeled with a specific task $l \in \ell$. This ensures that the encrypted and labeled data will not be used in training other models, even when the cloud and the shopper collude. To address **A3**, we introduce TTP to manage the master secret key rather than distribute it to the data shoppers for encryption, avoiding privacy inference for sellers' valuable raw data. Instead, we establish an independent pk for different data sellers to encrypt their data.

Apart from the new design goal proposed in this paper, the introduction of the labeled-FHIPE and TTP achieves those desirable security requirements in PPDE that have not been met in prior work. Our framework runs among TTP, data sellers, data shoppers, and a cloud. Our privacy-preserving data selection contains the following three steps:

Step 0 (Initialization): TTP initializes the system parameters and generates the pair of public key and master secret keys $(\mathsf{pk}, \mathsf{msk})$ and secret key sk for data seller.

Step 1 (Feed Forward): The data seller encrypts selling data $\mathbf{X}$ into $\mathsf{ct}_{\mathbf{X},l}$ using the label-FHIPE.Enc algorithm and task label l, then sends this to the cloud. Concurrently, the data shopper sends the masked first layer parameter matrix $\mathbf{Y}_1$ to TTP and generates decryption keys $\mathsf{dk}_{\mathbf{Y}_1,l}$. Parameters $\mathbf{W}_i$ or $\mathbf{k}_i$ of subsequent layers are encrypted into $\mathbf{RW}_i$ or $\mathbf{RK}_i$ by multiplying them with random matrices $\mathbf{R}_i$ for layers 2 through n, where n is the final layer number, and are then sent to the cloud. The cloud progressively computes $\mathbf{Z}_i$ under the mask to finally obtain the masked prediction values $\mathbf{Z}_n$.

Subsequent to the convolutional layer, a pooling layer is often introduced. Here, we consider a mean pooling layer with pooling window size of s_p, whose feature map, denoted as $\mathbf{P}_{i+1}$, is computed by feeding the output $\mathbf{Z}_i$ from the convolutional layer into it. The mean pooling layer can be computed as follows:

$$\mathbf{P}_{i+1}[j][k] = \frac{1}{s_p^2} \sum_{x=j \cdot s_p}^{i \cdot s_p + s_p - 1} \sum_{y=k \cdot s_p}^{k \cdot s_p + s_p - 1} \mathbf{Z}_i[x][y].$$

Step 2 (Data Selection): The data shopper receives $\mathbf{Z}_n$ and unmasks it by computing hadamard product $\mathbf{Z}_n \circ \mathbf{R}_i^{-2}$. After obtaining the original prediction values, the shopper then applies active learning to evaluate data informativeness.

In the privacy-preserving data validation, the shopper requires the cloud retrain its model with the selected data. Then, it can observe the prediction performance of the retrained model to estimate the quality of the selected data. The process

TABLE I
COMPUTATION COMPARISON

Model	Scheme	Data Enc. (s)	Model Enc. (s)	First Layer (s)	Sub. Layer (s)	Back Prop. (s)	Total (s)
Mnist	Primal	3.594	0	72.331	0.004	0.024	75.953
	Stanford	0.414	0.404	120.960	0.004	0.022	121.804
	Ours	**0.263 (13.67×)**	0.724	**6.981 (10.36×)**	0.004	0.024	**7.996 (9.50×)**
Cifar10	Primal	10.514	0	215.360	0.006	0.031	225.911
	Stanford	1.071	0.120	312.959	0.006	0.031	314.187
	Ours	**0.372 (27.30×)**	0.410	**19.243 (11.19×)**	0.006	0.033	**20.064 (11.26×)**
SVHN	Primal	9.978	0	211.127	0.006	0.035	221.146
	Stanford	1.171	0.114	311.456	0.005	0.037	312.783
	Ours	**0.312 (31.98×)**	0.420	**19.213 (10.99×)**	0.006	0.020	**19.971 (11.07×)**

of the privacy-preserving data validation includes an additional step, back propagation.

Step 3 (Back Propagation): We use the mean square error function as the cost function and computes the gradients of the last layer as $\delta = \mathbf{Z}_n^2 - \mathbf{L}$ for prediction value $\mathbf{Z}_n^2$ and labels $\mathbf{L}$. Then the cloud computes the gradients of each hidden layer and update all parameters according to randomized gradients δ'. Since the parameters of the first hidden layer and back layers are encrypted and masked by two methods, the cloud updates parameters in two forms.

Our complete framework is provided in Fig. 5.

VI. PERFORMANCE EVALUATION

A. Experimental Setup

Baseline. We set *Primal* [18] and *Stanford* as our baselines. In particular, *Primal* is the basic IPFE-based PPDE scheme but suffers from several security issues. *Stanford* is a scheme formed on the basis of our PPDE framework by replacing our labeled-FHIPE scheme with the FHIPE proposed by Stanford researchers in [26].

Dataset and Model Architecture. To evaluate the performance of our framework, we used two convolutional neural networks (CNNs) with publicly available datasets, MNIST [27], SVHN [28], and CIFAR10 [29].

Our Toy CNN (Model 1) is composed of two convolutional layers. The first layer contains 4 channels with 5×5 kernels and a stride of 2, followed by a 2×2 mean pooling operation with a stride of 2. The second layer contains 4 channels with 2×2 kernels and a stride of 1. A fully connected layer with 100×10 units and a Square activation function completes the network, bringing the total parameters to 1,182. An initial learning rate of 0.001 was used during training.

The network for the CIFAR10 and SVHN datasets, Model 2, consists of three convolutional layers. The first, second, and third layers contain 64, 128, and 256 channels, respectively, each with 3×3 kernels, a stride of 1, and padding of 1. Each layer is followed by a 2×2 mean pooling operation and a square activation layer. A fully connected layer with 4096×10 units and a square activation function is included, totaling $411,786$ parameters. Training this model involved common hyperparameters including an initial learning rate of 0.001 and a batch size of 64, following standard practices in the field.

Cryptosystem Implementation. Our cryptosystem is written in Go, based on the opensource functional encryption library GoFE [30]. Our implementation uses a hybrid design, where the underlying performance-intensive mathematical operations are implemented in go modules, i.e., the GoFE library, while the neural networks can be written in a readable, high-level language. Thus, we can implement different schemes in a unified platform for easier comparison.

Experimental Environment. We measure the computation time for two phases: Feed Forward Phase and Back Propagation Phase. We omit the initialization phase, which is considered a one-time cost. Our experiment was performed on a 64-bit Windows 10 laptop equipped with Intel Core i7-7700HQ CPU (2.80GHz) and 16GB RAM.

B. Precision Choice on Accuracy and Loss

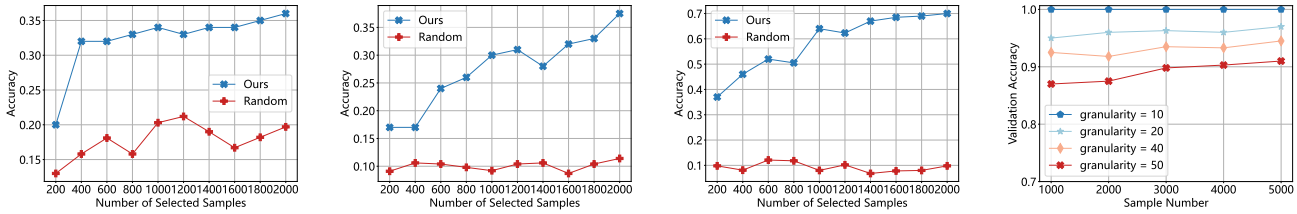
Since we use labeled-FHIPE in our approach, it only supports integer operations under ciphertext. We utilized our scheme to test the impact of different encryption precisions on accuracy and loss using three datasets, as shown in Fig. 6. To minimize the influence of encryption precision on the model during data evaluation, we chose an encryption precision of 14 (where seller data and shopper model parameters are multiplied by 2^{14} and rounded to the nearest integer) in the following experiments. As shown in Fig. 6, the model accuracy at this precision is nearly identical to that achieved under plaintext training. In the experiment, after an adequate number of training iterations, all three datasets have achieved an accuracy exceeding 90%. This verifies the performance of our models.

C. Computation Overhead

We compared the computational overhead of our approach with the Primal and Stanford schemes.

Since all the PPDE schemes employ FE encryption in the first layer and protect the subsequent layers using random masking techniques, we record the computational overhead for each scheme. This includes the cost of encrypting data by the data seller, the encryption cost of the model, the computation cost of the first layer, and the encryption and computation time for the subsequent layers. We tested different specifications of models with varying parameters in these layers to assess the impact on the computational overhead.

Overall Computation Costs. Table 1 documents the time expenditure of three strategies on three datasets and two models. Our proposed approach significantly reduces the overall

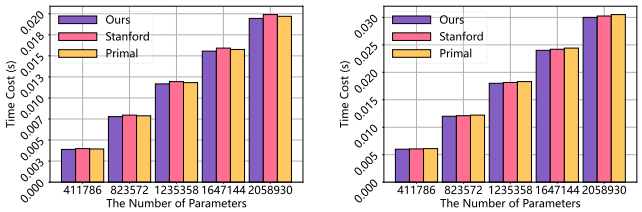


(a) Data Selection on Mnist (b) Data Selection on Cifar10 (c) Data Selection on SVHN (d) Data Validation

Fig. 8. Performance of Data Evaluation

computational cost by a factor of 10 compared to the state-of-the-art Primal strategy. Notably, our solution achieved considerable improvements in the most time-consuming aspects of the process—specifically data encryption and the efficiency of first-layer computation—with enhancements ranging from $13.67 \sim 31.98\times$ and $10.36 \sim 11.19\times$, respectively.

Data Encryption, Model Encryption, First Layer Computation. As depicted in Fig. 7, our method outperforms others in terms of time efficiency for seller encryption and first-layer computation. It's noteworthy that the first layer computations are the main time-consuming steps across all schemes. Yet, our scheme is the only one showing a linear time increase with the addition of parameters, demonstrating robust scalability, which is also mirrored in the model's precision. Despite higher computational overhead in Fig. 7(b) compared to Primal, it's crucial to remember that Primal does not perform model encryption, leaving privacy risks unaddressed.



(a) Subsequent Layer Enc. (b) Subsequent Layer Com.

Fig. 9. Comparison of Computational Overhead for Matrix Encryption and Computation

TABLE II
COMMUNICATION COMPARISON

Dataset	Entity	Primal (KB)	Stanford (KB)	Ours (KB)
Mnist	Shopper	3,085	53,077	108,166
	Data	519,535	489,955	253,764
Cifar10	Shopper	54,010,339	53,971,000	54,721,681
	Data	1,304,832	1,847,040	492,570
SVHN	Shopper	53,165,208	54,474,628	53,602,837
	Data	1,314,478	1,847,145	980,640

Matrix Encryption and Computation. We assess encryption and computation time for subsequent layers across two models and three schemes next. Fig. 9(a) and Fig. 9(b) show similar computational overhead growth across all schemes for both smaller (Model 1) and larger (Model 2) models.

In summary, our approach outperforms other approaches in computation efficiency for data sellers and the cloud, while the efficiency for shoppers is compromised due to first layer model encryption. This trade-off, essential for model security, might suggest Stanford's scheme as an alternative when dealing with limited computational resources.

D. Communication Overhead

We compared the main communication costs of the three schemes across three datasets and two different models, with an encryption precision of 14. From Tab. II, we can observe that our proposed scheme significantly reduces communication costs for data sellers, ranging from 1.34 to 2.05 times less compared to the state-of-the-art Primal scheme. However, the communication costs for data shoppers are relatively large. This is because our scheme encrypts the first layer of the model, resulting in ciphertext that is naturally larger than the non-ciphertext generated by Primal. We found that our scheme and the Stanford can complement each other in terms of communication costs for both sellers and shoppers.

E. Performance Analysis of Data Selection and Validation

We evaluated the performance of our privacy-preserving data selection method on three datasets and two networks. We recorded the performance of the first 2000 data samples using our data selection method compared to random data selection. Fig. 8(a-c) shows the performance of the two models on the three datasets after retraining. Our method consistently improved the model accuracy by more than 15% across all three datasets and two networks.

Regarding data validation, we obtained the same conclusion as the Primal, which is that smaller granularity leads to higher validation accuracy, as is shown in Fig. 8(d), but also results in larger computational overhead. Therefore, shoppers should make a trade-off between security and efficiency.

VII. CONCLUSION

This study highlights security shortcomings in the present FE-based PPDE scheme due to IPFE misuse and presents a redesigned PPDE strategy utilizing FE. We have introduced labeled-FHIPE, a novel FE variant, and developed a new PPFL framework that delegates master secret key management to TTP. Our scheme, tested on real-world datasets, has shown to outperform the current benchmark Primal by around $10\times$ and $2\times$ in computation and communication, respectively. It also promises flexible scalability and improved security for data evaluation services.

REFERENCES

[1] N. Sebe, *Machine learning in computer vision*. Springer Science & Business Media, 2005, vol. 29.

[2] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis *et al.*, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[3] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[4] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4101–4112, 2018.

[5] K. Ahiska, M. K. Ozgoren, and M. K. Leblebicioglu, "Autopilot design for vehicle cornering through icy roads," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 1867–1880, 2017.

[6] D. Acemoglu, A. Makhdoumi, A. Malekian, and A. Ozdaglar, "Too much data: Prices and inefficiencies in data markets," *American Economic Journal: Microeconomics*, vol. 14, no. 4, pp. 218–256, 2022.

[7] Dawex, "Data exchange, reveal the power of your data ecosystem," https://www.dawex.com/en/, accessed: July 5, 2023.

[8] Data Republic, "A platform for secure and trusted data exchange." https://www.datarepublic.com/, accessed: July 5, 2023.

[9] Z. Zheng, J. Zhu, and M. R. Lyu, "Service-generated big data and big data-as-a-service: an overview," in *2013 IEEE international congress on Big Data*. IEEE, 2013, pp. 403–410.

[10] J. Pei, "A survey on data pricing: from economics to data science," *IEEE Transactions on knowledge and Data Engineering*, vol. 34, no. 10, pp. 4586–4608, 2020.

[11] N. Hynes, D. Dao, D. Yan, R. Cheng, and D. Song, "A demonstration of sterling: a privacy-preserving data marketplace," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2086–2089, 2018.

[12] M. Travizano, C. Sarraute, M. Dolata, A. M. French, and H. Treiblmaier, "Wibson: A case study of a decentralized, privacy-preserving data marketplace," *Blockchain and distributed ledger technology use cases: Applications and lessons learned*, pp. 149–170, 2020.

[13] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1209–1222.

[14] M. Bakshi and M. Last, "Cryptornn-privacy-preserving recurrent neural networks using homomorphic encryption," in *Cyber Security Cryptography and Machine Learning: Fourth International Symposium, CSCML 2020, Be'er Sheva, Israel, July 2–3, 2020, Proceedings 4*. Springer, 2020, pp. 245–253.

[15] B. Reagen, W.-S. Choi, Y. Ko, V. T. Lee, H.-H. S. Lee, G.-Y. Wei, and D. Brooks, "Cheetah: Optimizing and accelerating homomorphic encryption for private inference," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 26–39.

[16] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," *arXiv preprint arXiv:2004.02229*, 2020.

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

[18] Q. Song, J. Cao, K. Sun, Q. Li, and K. Xu, "Try before you buy: Privacy-preserving data evaluation on cloud-based machine learning data marketplace," in *Annual Computer Security Applications Conference*, 2021, pp. 260–272.

[19] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," *Cryptology ePrint Archive*, 2015.

[20] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*. Springer, 2011, pp. 253–273.

[21] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, pp. 13–23.

[22] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *2009 ieee conference on computer vision and pattern recognition*. IEEE, 2009, pp. 2372–2379.

[23] R. Gay, "A new paradigm for public-key functional encryption for degree-2 polynomials," in *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part I*. Springer, 2020, pp. 95–120.

[24] C. Mouchet, J. Troncoso-Pastoriza, J.-P. Bossuat, and J.-P. Hubaux, "Multiparty homomorphic encryption from ring-learning-with-errors," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. CONF, pp. 291–311, 2021.

[25] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 89–98.

[26] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11*. Springer, 2018, pp. 544–562.

[27] C. C. Yann LeCun and B. Christopher, "Mnist handwritten digit database." http://yann.lecun.com/exdb/mnist/, 2010.

[28] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[29] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[30] F. Project, "gofe: A go library for functional encryption," https://github.com/fentec-project/gofe, 2022, accessed: 21-12-2023.